

HITTING AND HARVESTING PUMPKINS

GWENAËL JORET, CHRISTOPHE PAUL, IGNASI SAU,
SAKET SAURABH, AND STÉPHAN THOMASSÉ

ABSTRACT. The *c-pumpkin* is the graph with two vertices linked by $c \geq 1$ parallel edges. A *c-pumpkin-model* in a graph G is a pair $\{A, B\}$ of disjoint subsets of vertices of G , each inducing a connected subgraph of G , such that there are at least c edges in G between A and B . We focus on covering and packing *c-pumpkin-models* in a given graph: On the one hand, we provide an FPT algorithm running in time $2^{O(k)}n^{O(1)}$ deciding, for any fixed $c \geq 1$, whether all *c-pumpkin-models* can be covered by at most k vertices. This generalizes known single-exponential FPT algorithms for VERTEX COVER and FEEDBACK VERTEX SET, which correspond to the cases $c = 1, 2$ respectively. On the other hand, we present a $\mathcal{O}(\log n)$ -approximation algorithm for both the problems of covering all *c-pumpkin-models* with a smallest number of vertices, and packing a maximum number of vertex-disjoint *c-pumpkin-models*.

Keywords: Covering and packing; parameterized complexity; approximation algorithm; single-exponential algorithm; iterative compression; graph minors.

1. INTRODUCTION

The *c-pumpkin* is the graph with two vertices linked by $c \geq 1$ parallel edges. A *c-pumpkin-model* in a graph G is a pair $\{A, B\}$ of disjoint subsets of vertices of G , each inducing a connected subgraph of G , such that there are at least c edges in G between A and B . In this article we study the problems of covering all *c-pumpkin-models* of a given graph G with few vertices, and packing as many disjoint *c-pumpkin-models* in G as possible. As discussed below, these problems generalize several well-studied problems in algorithmic graph theory. We focus on FPT algorithms for the parameterized version of the covering problem, as well as on poly-time approximation algorithms for the optimization version of both the packing and covering problems.

FPT algorithms. From the parameterized complexity perspective, we study the following problem for every fixed integer $c \geq 1$.

p-c-PUMPKIN-COVERING (*p-c-COVER* for short)
Instance: A graph G and a non-negative integer k .
Parameter: k .
Question: Does there exist $S \subseteq V(G)$, $|S| \leq k$, such that $G \setminus S$ does not contain the *c-pumpkin* as a minor?

When $c = 1$, the *p-c-COVER* problem is the *p-VERTEX COVER* problem [2, 9]. For $c = 2$, it is the *p-FEEDBACK VERTEX SET* problem [7, 11, 20]. When $c = 3$, this corresponds to the recently introduced *p-DIAMOND HITTING SET* problem [15].

The *p-c-COVER* problem can also be seen as a particular case of the following problem, recently introduced by Fomin *et al.* [18] and studied from the kernelization perspective:

This work was supported in part by AGAPE (ANR-09-BLAN-0159) and GRATOS (ANR-09-JCJC-0041-01) projects (France), by a project funded by DAE (India), and by the Actions de Recherche Concertées (ARC) fund of the Communauté française de Belgique (Belgium). Gwenaël Joret is a Post-doctoral Researcher of the Fonds National de la Recherche Scientifique (F.R.S.-FNRS).

Let \mathcal{F} be a finite set of graphs. In the $p\text{-}\mathcal{F}\text{-COVER}$ problem, we are given an n -vertex graph G and an integer k as input, and asked whether at most k vertices can be deleted from G such that the resulting graph does not contain any graph from \mathcal{F} as a minor. Among other results, it is proved in [18] that if \mathcal{F} contains a c -pumpkin for some $c \geq 1$, then $p\text{-}\mathcal{F}\text{-COVER}$ admits a kernel of size $\mathcal{O}(k^2 \log^{3/2} k)$. As discussed in Section 3, this kernel leads to a simple FPT algorithm for $p\text{-}\mathcal{F}\text{-COVER}$ in this case, and in particular for $p\text{-}c\text{-COVER}$, with running time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$. A natural question is whether there exists an algorithm for $p\text{-}c\text{-COVER}$ with running time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ for every fixed $c \geq 1$. Such algorithms are called *single-exponential*. For the $p\text{-VERTEX COVER}$ problem the existence of single-exponential algorithms is well-known since almost the beginnings of the field of Parameterized Complexity [2], the best current algorithm being by Chen *et al.* [9]. On the other hand, the question about the existence of single-exponential algorithms for $p\text{-FEEDBACK VERTEX SET}$ was open for a long time and was finally settled independently by Guo *et al.* [20] (using iterative compression) and by Dehne *et al.* [11]. The current champion algorithm for $p\text{-FEEDBACK VERTEX SET}$ runs in time $\mathcal{O}(3.83^k k \cdot n^2)$ [7].

We present in Section 3 a single-exponential algorithm for $p\text{-}c\text{-COVER}$ for every fixed $c \geq 1$, using a combination of a kernelization-like technique and iterative compression. Notice that this generalizes the above results for $p\text{-VERTEX COVER}$ and $p\text{-FEEDBACK VERTEX SET}$. We remark that asymptotically these algorithms are optimal, that is, it is known that unless ETH fails neither $p\text{-VERTEX COVER}$ nor $p\text{-FEEDBACK VERTEX SET}$ admit an algorithm with running time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ [8, 21]. It is worth to mention here that a similar quantitative approach was taken by Lampis [23] for graph problems expressible in MSOL parameterized by the sum of the formula size and the size of a minimum vertex cover of the input graph.

Approximation algorithms. For a fixed integer $c \geq 1$, we define the following two optimization problems.

MINIMUM c -PUMPKIN-COVERING (MIN c -COVER for short)

Input: A graph G .

Output: A subset $S \subseteq V(G)$ such that $G \setminus S$ does not contain the c -pumpkin as a minor.

Objective: Minimize $|S|$.

MAXIMUM c -PUMPKIN-PACKING (MAX c -PACKING for short)

Input: A graph G .

Output: A collection \mathcal{M} of vertex-disjoint subgraphs of G , each containing the c -pumpkin as a minor.

Objective: Maximize $|\mathcal{M}|$.

Let us now discuss how the above problems encompass several well-known problems. For $c = 1$, MIN 1-COVER is the MINIMUM VERTEX COVER problem, which can be easily 2-approximated by finding any maximal matching, whereas MAX 1-PACKING corresponds to finding a MAXIMUM MATCHING, which can be done in polynomial time. For $c = 2$, MIN 2-COVER is the MINIMUM FEEDBACK VERTEX SET problem, which can be also 2-approximated [1, 3], whereas MAX 2-PACKING corresponds to MAXIMUM VERTEX-DISJOINT CYCLE PACKING, which can be approximated to within a $\mathcal{O}(\log n)$ factor [22]. For $c = 3$, MIN 3-COVER is the DIAMOND HITTING SET problem studied by Fiorini *et al.* in [15], where a 9-approximation algorithm is given.

We provide in Section 4 an algorithm that approximates both the MIN c -COVER and the MAX c -PACKING problems to within a $\mathcal{O}(\log n)$ factor for every fixed $c \geq 1$. Note that this algorithm matches the best existing algorithms for MAX c -PACKING for $c = 2$ [22]. For the MIN c -COVER problem, our result is only a slight improvement on the $\mathcal{O}(\log^{3/2} n)$ -approximation algorithm given in [18]. However, for the MAX c -PACKING problem, there was no approximation algorithm known before except for the $c \leq 2$ case. Also, let us remark that, for $c \geq 2$ and every fixed $\varepsilon > 0$, MAX c -PACKING is quasi-NP-hard to approximate to within a $\mathcal{O}(\log^{1/2-\varepsilon} n)$ factor: For $c = 2$ this was shown by Friggstad and Salavatipour [19], and their result can be extended to the case $c > 2$ in the following straightforward way: given an instance G of MAX 2-PACKING, we build an instance of MAX c -PACKING by replacing each edge of G with $c - 1$ parallel edges.

The main ingredient of our approximation algorithm is the following combinatorial result: We show that every n -vertex graph G either contains a small c -pumpkin-model or has a structure that can be reduced in polynomial time, giving a smaller equivalent instance for both the MIN c -COVER and the MAX c -PACKING problems. Here by a “small” c -pumpkin-model, we mean a model of size at most $f(c) \cdot \log n$ for some function f independent of n . This result extends one of Fiorini *et al.* [15], who dealt with the case $c = 3$.

2. PRELIMINARIES

Graphs. We use standard graph terminology, see for instance [12]. All graphs in this article are finite and undirected, and may have parallel edges but no loops. We will sometimes restrict our attention to simple graphs, that is, graphs without parallel edges.

Given a graph G , we denote the vertex set of G by $V(G)$ and the edge set of G by $E(G)$. We use the shorthand $|G|$ for the number of vertices in G . For a subset $X \subseteq V(G)$, we use $G[X]$ to denote the subgraph of G induced by X . For a subset $Y \subseteq E(G)$ we let $G[Y]$ be the graph with $E(G[Y]) := Y$ and with $V(G[Y])$ being the set of vertices of G incident with some edge in Y . For a subset $X \subseteq V(G)$, we may use the notation $G \setminus X$ to denote the graph $G[V(G) \setminus X]$.

The set of neighbors of a vertex v of a graph G is denoted by $N_G(v)$. The *degree* $\deg_G(v)$ of a vertex $v \in V(G)$ is defined as the number of edges incident with v (thus parallel edges are counted). We write $\deg_G^*(v)$ for the number of neighbors of v , that is, $\deg_G^*(v) := |N_G(v)|$. Similarly, given a subgraph $H \subseteq G$ with $v \in V(H)$, we can define in the natural way $N_H(v)$, $\deg_H(v)$, and $\deg_H^*(v)$, that is, $N_H(v) = N_G(v) \cap V(H)$, $\deg_H(v)$ is the number of edges incident with v with both endpoints in H , and $\deg_H^*(v) = |N_H(v)|$. In these notations, we may drop the subscript if the graph is clear from the context. The minimum degree of a vertex in a graph G is denoted $\delta(G)$, and the maximum degree of a vertex in G is denoted $\Delta(G)$. We use the notation $\mathbf{cc}(G)$ to denote the number of connected components of G . Also, we let $\mu(G)$ denote the maximum multiplicity of an edge in G .

Minors and models. Given a graph G and an edge $e \in E(G)$, let $G \setminus e$ be the graph obtained from G by removing the edge e , and let G/e be the graph obtained from G by contracting e (we note that parallel edges resulting from the contraction are kept but loops are deleted). If H can be obtained from a subgraph of G by a (possibly empty) sequence of edge contractions, we say that H is a *minor* of G , and we denote it by $H \preceq_m G$. A graph G is *H -minor-free*, or simply *H -free*, if G does not contain H as a minor. A *model* of a graph H , or simply *H -model*, in a graph G is a collection $\{S_v \subseteq V(G) \mid v \in V(H)\}$ such that

- (i) $G[S_v]$ is connected for every $v \in V(H)$;

- (ii) S_v and S_w are disjoint for every two distinct vertices v, w of H ; and
- (iii) there are at least as many edges between S_v and S_w in G as between v and w in H , for every $vw \in E(H)$.

The *size* of the model is defined as $\sum_{v \in V(H)} |S_v|$. Clearly, H is a minor of G if and only if there exists a model of H in G . In this paper, we will almost exclusively consider H -models with H being isomorphic to a c -pumpkin for some $c \geq 1$. Thus a c -pumpkin-model in a graph G is specified by an unordered pair $\{A, B\}$ of disjoint subsets of vertices of G , each inducing a connected subgraph of G , such that there are at least c edges in G between A and B . A c -pumpkin-model $\{A, B\}$ of G is said to be *minimal* if there is no c -pumpkin-model $\{A', B'\}$ of G with $A' \subseteq A$, $B' \subseteq B$, and $|A'| + |B'| < |A| + |B|$.

A subset X of vertices of a graph G such that $G \setminus X$ has no c -pumpkin-minor is called a *c-pumpkin-hitting set*, or simply *c-hitting set*. We denote by $\tau_c(G)$ the minimum size of a c -pumpkin-hitting set in G . A collection \mathcal{M} of vertex-disjoint subgraphs of a graph G , each containing a c -pumpkin-model, is called a *c-pumpkin-packing*, or simply *c-packing*. We denote by $\nu_c(G)$ the maximum size of a c -pumpkin-packing in G . Obviously, for any graph G it holds that $\nu_c(G) \leq \tau_c(G)$, but the converse is not necessarily true.

The following lemma on models will be useful in our algorithms. The proof is straightforward and hence is omitted.

Lemma 2.1. *Suppose G' is obtained from a graph G by contracting some vertex-disjoint subgraphs of G , each of diameter at most k . Then, given an H -model in G' of size s , one can compute in polynomial time an H -model in G of size at most $k \cdot \Delta(H) \cdot s$.*

Parameterized algorithms. See for instance [13] for an introduction to Parameterized Complexity. A *parameterized problem* Π is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet Γ . An instance of a parameterized problem consists of a pair (x, k) , where k is called the *parameter*. A central notion in parameterized complexity is *fixed parameter tractability (FPT)*, which means, for a given instance (x, k) , solvability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial in the input size.

A *kernelization algorithm* or, in short, a *kernel* for a parameterized problem $\Pi \subseteq \Gamma^* \times \mathbb{N}$ is an algorithm that given $(x, k) \in \Gamma^* \times \mathbb{N}$ outputs in time polynomial in $|x| + k$ a pair $(x', k') \in \Gamma^* \times \mathbb{N}$ such that

- (i) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$; and
- (ii) $\max\{|x'|, k'\} \leq g(k)$,

where g is some computable function. The function g is referred to as the *size* of the kernel. If $g(k) = k^{O(1)}$ or $g(k) = O(k)$, then we say that Π admits a polynomial kernel and a linear kernel, respectively.

Iterative compression is a tool that has been used successfully in finding fast FPT algorithms for a number of parameterized problems. The main idea behind iterative compression is an algorithm which, given a solution of size $k + 1$ for a problem, either compresses it to a solution of size k or proves that there is no solution of size k . This technique was first introduced by Reed *et al.* to solve the ODD CYCLE TRANSVERSAL problem [25], where one is interested in finding a set of at most k vertices whose deletion makes the graph bipartite [25]. Since then, it has been extensively used in the literature, see for instance [10, 17, 20].

Tree-width. We briefly recall the definition of the tree-width of a graph. A *tree decomposition* of a graph G is an ordered pair $(T, \{W_x \mid x \in V(T)\})$ where T is a tree and $\{W_x \mid x \in V(T)\}$ a family of subsets of $V(G)$ (called *bags*) such that

- (i) $\bigcup_{x \in V(T)} W_x = V(G)$;
- (ii) for every edge $uv \in E(G)$, there exists $x \in V(T)$ with $u, v \in W_x$; and

(iii) for every vertex $u \in V(G)$, the set $\{x \in V(T) \mid u \in W_x\}$ induces a subtree of T . The *width* of tree decomposition $(T, \{W_x \mid x \in V(T)\})$ is $\max\{|W_x| - 1 \mid x \in V(T)\}$. The *tree-width* $\text{tw}(G)$ of G is the minimum width among all tree decompositions of G . We refer the reader to Diestel's book [12] for an introduction to the theory of tree-width. It is an easy exercise to check that the tree-width of a simple graph is an upper bound on its minimum degree. This implies the following lemma.

Lemma 2.2. *Every n -vertex simple graph with tree-width k has at most $k \cdot n$ edges.*

We will need the following result of Bodlaender *et al* [6].

Theorem 2.3 (Bodlaender *et al.* [6]). *Every graph not containing a c -pumpkin as a minor has tree-width at most $2c - 1$.*

The following corollary is an immediate consequence of the above theorem.

Corollary 2.4. *Every n -vertex graph with no minor isomorphic to a c -pumpkin has at most $(c - 1) \cdot (2c - 1) \cdot n$ edges.*

3. A SINGLE-EXPONENTIAL FPT ALGORITHM

As mentioned in the introduction, it is proved in [18] that given an instance (G, k) of p - \mathcal{F} -COVER such that \mathcal{F} contains a c -pumpkin for some $c \geq 1$, one can obtain in polynomial time an equivalent instance of size $\mathcal{O}(k^2 \log^{3/2} k)$. This kernel leads to the following simple FPT algorithm for p - \mathcal{F} -COVER: First compute the kernel K in polynomial time, and then for every subset $S \subseteq V(K)$ of size k , test whether $K[V(K) \setminus S]$ contains any of the graphs in \mathcal{F} as a minor, using the poly-time algorithm of Robertson and Seymour [26] for every graph in \mathcal{F} . If for some S we have that $K[V(K) \setminus S]$ contains no graph from \mathcal{F} as a minor, we answer YES; otherwise the answer is NO. The running time of this algorithm is clearly bounded by $\binom{k^2 \log^{3/2} k}{k} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$.

In this section we give an algorithm for p -c-PUMPKIN-COVERING that runs in time $d^k \cdot n^{\mathcal{O}(1)}$ for any fixed $c \geq 1$, where d only depends on the fixed constant c . Towards this, we first introduce a variant of p -c-PUMPKIN-COVERING, namely p -c-PUMPKIN-DISJOINT COVERING, or p -c-DISJOINT COVER for short. In p -c-DISJOINT COVER, apart from a graph G and a positive integer k , we are also given a set S of size at most $k + 1$ such that $G \setminus S$ does not contain the c -pumpkin as a minor. Here the objective is to find a set $S' \subseteq V(G) \setminus S$ such that $|S'| \leq k$ and $G \setminus S'$ does not contain the c -pumpkin as a minor. Next we show a lemma that allows us to relate the two problems mentioned above.

Lemma 3.1. *If p -c-DISJOINT COVER can be solved in time $\eta(c)^k \cdot n^{\mathcal{O}(1)}$, then p -c-COVER can be solved in time $(\eta(c) + 1)^k \cdot n^{\mathcal{O}(1)}$.*

Proof. We order the vertices of $V(G)$ as v_1, v_2, \dots, v_n and define $V_i = \{v_1, \dots, v_i\}$ for every $1 \leq i \leq n$. Notice that if G has a c -hitting set S of size k then $S \cap V_i$ is a c -hitting set of $G[V_i]$ for every $i \leq n$. Furthermore, if S is a c -hitting set of $G[V_i]$ then $S \cup \{v_{i+1}\}$ is a c -hitting set of $G[V_{i+1}]$. Finally, V_k is a c -hitting set of size k of $G[V_k]$. These three facts together with the $\eta(c)^k \cdot n^{\mathcal{O}(1)}$ algorithm for p -c-DISJOINT COVER give a $(\eta(c) + 1)^k \cdot n^{\mathcal{O}(1)}$ time algorithm for p -c-COVER as follows.

Given $(G[V_{k+1}], V_{k+1}, k)$ we guess the intersection of our prospective S' with V_{k+1} and recurse. That is, for every subset $Y \subseteq V_{k+1}$ such that $|Y| \leq k$ we want to check whether there exists a set $Z \subseteq V_{k+1} \setminus Y$ such that $Y \cup Z$ is a c -hitting set of size at most k for $G[V_{i+1}]$. We first check whether $G[V_{k+1} \setminus Y]$ is c -pumpkin-free; if it is then we get an instance $(G[V_{k+1} \setminus Y], V_{k+1} \setminus Y, k - |Y|)$ of p -c-DISJOINT COVER. On this instance we call an algorithm for p -c-DISJOINT COVER. If for every Y the algorithm returns that $(G[V_{k+1} \setminus Y], N, k - |Y|)$ has no c -hitting set of size at most k , then we return that G

has no k -sized c -hitting set. Else there exists a set Y for which the algorithm returns a c -hitting set Z of size at most $k - |Y|$, call $Z \cup Y$ as S_{k+1} . Observe that the running time of this step is bounded by $\sum_{i=0}^k \binom{k+1}{i} (\eta(c) + 1)^{k-i} \cdot n^{\mathcal{O}(1)} = (\eta(c) + 1)^k \cdot n^{\mathcal{O}(1)}$.

We repeat the steps described in the second paragraph, now with input $(G[V_{k+2}], S_{k+1} \cup \{v_{k+2}\}, k)$. Again, we either receive a “no” answer or a k -sized c -hitting set S_{k+2} of $G[V_{k+2}]$ and again, if the answer is negative then G has no k -sized c -hitting set. Otherwise we call the algorithm described in the second paragraph with input $(G, S_{k+2} \cup \{v_{k+3}\}, k)$ and keep going on in a similar manner. If we receive a negative answer at some step we answer that G has no k -sized c -hitting set. If we do not receive a negative answer at any step, then after $n - k$ rounds we have a k -sized c -hitting set of $G[V_n] = G$. Thus we have resolved the input instance in time $(\eta(c) + 1)^k \cdot n^{\mathcal{O}(1)}$. \square

Lemma 3.1 allows us to focus on p - c -DISJOINT COVER. In what follows we give an algorithm for p - c -DISJOINT COVER that runs in single-exponential time.

Overview of the algorithm. The algorithm for p - c -DISJOINT COVER is based on a combination of branching and poly-time preprocessing. Let (G, S, k) be the given instance of p - c -DISJOINT COVER and let $V := V(G)$. Our main objective is to eventually show that $A := \{v \in V \setminus S : N_G(v) \cap S \neq \emptyset\}$ has cardinality $\mathcal{O}(k)$. As far as we cannot guarantee this upper bound, we will see that we can branch suitably to obtain a few subproblems. Once we have the desired upper bound, we use a protrusion-based reduction rule from [18] to give a poly-time procedure that given an instance (G, S, k) of p - c -DISJOINT COVER, returns an equivalent instance (G', S, k') such that G' has $\mathcal{O}(k)$ vertices. That is, we obtain a linear vertex kernel for p - c -DISJOINT COVER in this particular case. Notice that once we have a linear vertex kernel of size αk for p - c -DISJOINT COVER, we can solve the problem in $\binom{\alpha k}{k} \cdot k^{\mathcal{O}(1)}$. So the overall algorithm consists of a recursion tree where in leaf nodes we obtain a linear kernel and then solve the problem using brute force enumeration.

We can now proceed to the formal description of the algorithm. Let

$$\begin{aligned} V_1 &:= \{v \in V \setminus S : |N_G(v) \cap S| = 1\} \\ V_{\geq 2} &:= \{v \in V \setminus S : |N_G(v) \cap S| \geq 2\}. \end{aligned}$$

Linear kernel. We start of with a procedure, called *protrusion rule*, that bounds the size of our graph when $|A = V_1 \cup V_{\geq 2}| = \mathcal{O}(k)$. Given $R \subseteq V(G)$, we define $\partial_G(R)$ as the set of vertices in R that have a neighbor in $V(G) \setminus R$. Thus the neighborhood of R is $N_G(R) = \partial_G(V(G) \setminus R)$. We say that a set $X \subseteq V(G)$ is an r -*protrusion* of G if $\mathbf{tw}(G[X]) \leq r$ and $|\partial_G(X)| \leq r$. We now state the protrusion rule.

P Let (G, S, k) be an instance and let $\gamma : \mathbb{N} \rightarrow \mathbb{N}$ be the function defined in [18, Lemma 5] (originally shown in [4]). If G contains a $4c$ -protrusion Y of size at least $\gamma(4c)$, then replace Y to obtain an equivalent instance (G^*, S, k^*) such that $|V(G^*)| < |V(G)|$.

The proof of the next lemma is identical to the proof of [18, Theorem 1], we provide it here for the sake of completeness.

Lemma 3.2. *If $|A = V_1 \cup V_{\geq 2}| = \mathcal{O}(k)$, then p - c -DISJOINT COVER has a kernel with $\mathcal{O}(|A|) = \mathcal{O}(k)$ vertices.*

Proof. Let (G, S, k) be an instance of p - c -DISJOINT COVER such that $|A|$ is bounded by $\mathcal{O}(k)$. We apply [18, Lemma 4] and obtain in linear time a $2((2c - 1) + 1) = 4c$ -protrusion Y of G of size at least $\frac{|V(G)| - |S|}{4|A| + 1}$ in $G \setminus S$. Let $\gamma : \mathbb{N} \rightarrow \mathbb{N}$ be the function defined in [18, Lemma 5] (originally shown in [4]). If $\frac{|V(G)| - |S|}{4|A| + 1} \geq \gamma(4c)$, then using [18,

Lemma 5] we replace the $4c$ -protrusion Y in G and obtain an instance (G^*, S, k^*) such that $|V(G^*)| < |V(G)|$, $k^* \leq k$, and (G^*, S, k^*) is a YES-instance of p - c -DISJOINT COVER if and only if (G, S, k) is a YES-instance of p - c -DISJOINT COVER. In the whole process we never delete either the vertices of S or A .

Let (G^*, S, k^*) be the reduced instance. In other words, there is no $4c$ -protrusion of size $\gamma(4c)$ in $G^* \setminus S$, and protrusion rule **P** no longer applies. We claim that the number of vertices in this graph is bounded by $\mathcal{O}(k)$. Indeed, since we cannot apply protrusion rule **P**, we have that $\frac{|V(G^*)| - |S|}{4|A| + 1} \leq \gamma(4c)$. Because $k^* \leq k$, we have that $|V(G^*)| \leq \gamma(4c)(4|A| + 1) + |S|$. Since $|A| = \mathcal{O}(k)$ and $|S| \leq k$ we have that $|V(G^*)| = \mathcal{O}(k)$. This completes the proof. \square

Branching procedure. For our branching algorithm we take the following measure:

$$(1) \quad \mu = \mathbf{cc}(G[S]) + k.$$

For simplicity we call the vertices in V_1 *white*. We start with some simple reduction rules (depending on c) that will be applied in the compression routine. We also prove, together with the description of each rule, that they are valid for our problem:

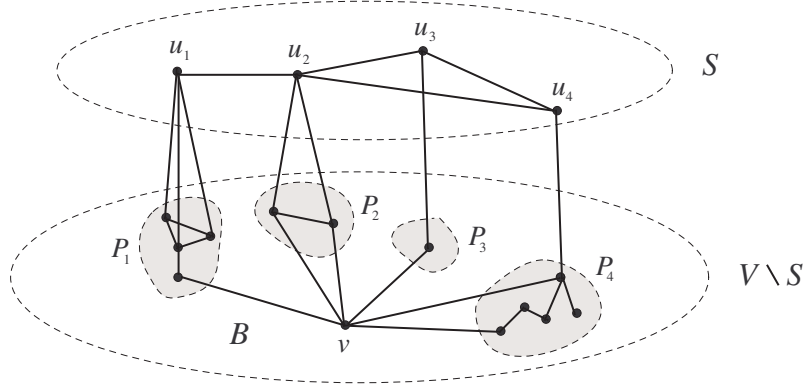
- R1** Let C be a connected component of $G[V \setminus S]$. If C does not have any neighbor in S , we can safely delete C , as its vertices will never participate in a c -pumpkin-model.
- R2** Let C be a connected component of $G[V \setminus (S \cup \{v\})]$ for some vertex $v \in V \setminus S$. If C does not have any neighbor in S , we can safely delete C , as its vertices will never participate in a minimal c -pumpkin-model.
- R3** Let C be a connected component of $G[V \setminus S]$. If C has exactly one neighbor v in S and $G[V(C) \cup \{v\}]$ is c -pumpkin-free, we can safely delete it, as its vertices will never participate in a minimal c -pumpkin-model.
- R4** Let B be a connected induced subgraph (not necessarily a connected component) of $G[V \setminus S]$, let $v \in V(B)$, and let P_1, \dots, P_ℓ be the connected components of $G[V(B) \setminus \{v\}]$. Assume that $\ell \geq c$ and that the following conditions hold:
 - (i) For $1 \leq i \leq \ell$, no vertex in P_i has a neighbor in $V \setminus (S \cup V(P_i) \cup \{v\})$.
 - (ii) For $1 \leq i \leq \ell$, there exists a vertex $u_i \in S$ such that $\cup_{w \in V(P_i)} N_G(w) \cap S = \{u_i\}$.
 - (iii) For $1 \leq i \leq \ell$, $G[V(P_i) \cup \{u_i\}]$ is c -pumpkin-free.
 - (iv) The vertices u_1, \dots, u_ℓ belong to the same connected component D of $G[S]$.

Then we include vertex v in the solution and we decrease the parameter by one. Indeed, note that as $\ell \geq c$ and by conditions (ii) and (iv), $G[V(B) \cup V(D)]$ contains a c -pumpkin, so any solution needs to contain at least one vertex of B , since we are looking for a solution that does not include vertices from S . On the other hand, by condition (iii) every minimal c -pumpkin-model intersecting B necessarily contains vertex v , and by condition (i) the deletion of v guarantees that no minimal c -pumpkin-model of $G \setminus \{v\}$ contains a vertex of B . Therefore, we can safely include vertex v in the solution and decrease the parameter accordingly. See Fig. 1 for an illustration for $c = 4$.

We say that the instance (G, S, k) is (S, c) -*reduced* if rules **R1**, **R2**, **R3**, or **R4** cannot be applied anymore. Note that these reduction rules can easily be applied in polynomial time.

We now describe a branching rule that will be used in our compression routine:

- B** let P be a simple path in $G[V \setminus S]$ with $|V(P)| = \ell$ and let v_1 and v_2 be the endpoints of P . Suppose that v_1 (resp. v_2) has a neighbor in a connected component C_1 (resp. C_2) of $G[S]$, with $C_1 \neq C_2$. Then we branch for all 2^ℓ

FIGURE 1. Illustration of reduction rule **R4** for $c = 4$.

subsets of vertices in P . Note that in every case we decrease the measure of the iterative compression, as either we include at least one vertex in the solution or we decrease the number of connected components of $G[S]$.

The branching rule above will be used on paths P that are “small enough”. The following Lemmas 3.3 and 3.6 are key to our algorithm. We also need two intermediate technical results, which will be used in the proof of Lemma 3.6.

Lemma 3.3. *There is a function $f(c)$ such that if (G, S, k) is an (S, c) -reduced YES-instance to the p -c-DISJOINT COVER problem, then $|V_{\geq 2}| \leq f(c) \cdot k$.*

Proof. In order to upper-bound $|V_{\geq 2}|$, we build from $G[S]$ the following auxiliary graph H : we start with $H = (S, E(G[S]))$, and for each vertex $v \in V_{\geq 2}$ with neighbors u_1, \dots, u_ℓ , $\ell \geq 2$, we add to H an edge e_v between two arbitrary neighbors u_1, u_2 of v . Note that $H \preceq_m G$, and that for each vertex $v \in V_{\geq 2}$, $H \setminus e_v \preceq_m G \setminus v$. We now argue that $|E(H)|$ is linearly bounded by k , which implies the desired result as by construction $|V_{\geq 2}| \leq |E(H)|$. If G is a YES-instance, there must be a set $S' \subseteq V \setminus S$, $|S'| \leq k$, such that $G \setminus S'$ is c -pumpkin-free. By construction of H , the removal of each vertex $v \in S' \cap V_{\geq 2}$ corresponds to the removal of the edge $e_v \in E(H)$. Let $H' = \{H \setminus e_v : v \in S' \cap V_{\geq 2}\}$, and note that $H' \preceq_m G \setminus S'$, so H' is c -pumpkin-free. Therefore, by Corollary 2.4 it follows that $|E(H')| \leq (c-1) \cdot (2c-1) \cdot |V(H')| \leq (c-1) \cdot (2c-1) \cdot (k+1)$. As $|E(H)| \leq |E(H')| + k$, we conclude that $|V_{\geq 2}| \leq |E(H)| \leq (c-1) \cdot (2c-1) \cdot (k+1) + k$. \square

Lemma 3.4. *There is a function $g(c)$ such that if (G, S, k) is an (S, c) -reduced YES-instance to the p -c-DISJOINT COVER problem and \mathcal{C} is a collection of disjoint connected subgraphs of $G[V \setminus S]$ such that each subgraph has at least two distinct neighbors in S , then $|\mathcal{C}| \leq g(c) \cdot k$.*

Proof. The proof is very similar to the proof of Lemma 3.3. In order to upper-bound $|\mathcal{C}|$, we build from $G[S]$ the following auxiliary graph H : we start with $H = (S, E(G[S]))$, and for each subgraph $C \in \mathcal{C}$ with neighbors u_1, \dots, u_ℓ , $\ell \geq 2$, we add to H an edge e_C between two arbitrary neighbors u_1, u_2 of C . Note that $H \preceq_m G$, and that for each subgraph $C \in \mathcal{C}$, $H \setminus e_C \preceq_m G \setminus C$. We now argue that $|E(H)|$ is linearly bounded by k , which implies the desired result as by construction $|\mathcal{C}| \leq |E(H)|$. If G is a YES-instance, there must be a set $S' \subseteq V \setminus S$, $|S'| \leq k$, such that $G \setminus S'$ is c -pumpkin-free. By construction of H , the removal of a vertex v in a subgraph $C \in \mathcal{C}$ corresponds to the removal of at most an edge $e_C \in E(H)$ (as maybe the edge e_C can still be *simulated* after the removal of v). Let H' be the subgraph obtained from H after the removal of those edges, and note that $H' \preceq_m G \setminus S'$, so H' is c -pumpkin-free. Therefore, by Corollary 2.4

it follows that $|E(H')| \leq (c-1) \cdot (2c-1) \cdot |V(H')| \leq (c-1) \cdot (2c-1) \cdot (k+1)$. As $|E(H)| \leq |E(H')| + k$, we conclude that $|C| \leq |E(H)| \leq (c-1) \cdot (2c-1) \cdot (k+1) + k$. \square

Lemma 3.5. *In an (S, c) -reduced instance (G, S, k) , the number of connected components of $G[V \setminus S]$ is $\mathcal{O}(k)$.*

Proof. Note that by reduction rules **R1** and **R3**, we can assume that each connected component C of $G[V \setminus S]$ has some neighbor in S , and that if C has exactly one neighbor v in S , then $G[V(C) \cup \{v\}]$ has a c -pumpkin. We now distinguish two types of connected components of $G[V \setminus S]$:

- The number of components C that have exactly one neighbor v in S and such that $G[V(C) \cup \{v\}]$ contains the c -pumpkin as a minor is at most k , as any solution needs to contain at least one vertex from each such connected component.
- The number of components that have at least two neighbors in S is $\mathcal{O}(k)$ by Lemma 3.4.

This concludes the proof. \square

Now we prove our key structural lemma.

Lemma 3.6. *There is a function $h(c)$ such that if (G, S, k) is an (S, c) -reduced YES-instance to the p -c-DISJOINT COVER problem, then either $|V_1| \leq h(c) \cdot k$, or we can apply protrusion rule **P**, or we can branch according to branching rule **B**.*

Proof. We proceed to find a packing of disjoint connected subgraphs $\mathcal{P} = \{B_1, \dots, B_\ell\}$ of $G[V \setminus S]$ containing all white vertices except for $\mathcal{O}(k)$ of them. This will help us in bounding $|V_1|$. We call the subgraphs in \mathcal{P} *blocks*. For a graph $H \subseteq G[V \setminus S]$, let $w(H)$ be the number of white vertices in H . The idea is to obtain, as far as possible, blocks B with $c \leq w(B) \leq c^3$; we call these blocks *suitable*, and the other blocks are called *non-suitable*. If at some point we cannot refine the packing anymore in order to obtain suitable blocks, we will argue about its structural properties, that will allow us to either bound the number of white vertices, or to apply protrusion rule **P**, or to branch according to branching rule **B**.

We start with \mathcal{P} containing all the connected components C of $G[V \setminus S]$ such that $w(C) > c^3$, and we recursively try to refine the current packing. By Lemma 3.5, we know that the number of connected components is $\mathcal{O}(k)$, and hence the number of white vertices, that are not included in \mathcal{P} is $\mathcal{O}(c^3 k) = \mathcal{O}(k)$.

More precisely, for each block B with $w(B) > c^3$, we build a spanning tree T of B , and we orient each edge $e \in E(T)$ towards the components of $T \setminus \{e\}$ containing at least c white vertices. (Note that, as $w(B) > c^3$, each edge gets at least one orientation, and that edges may be oriented in both directions.) If some edge $e \in E(T)$ is oriented in both directions, we replace in \mathcal{P} block B with the subgraphs induced by the vertices in each of the two subtrees. We stop this recursive procedure whenever we cannot find more suitable blocks using this orientation trick. Let \mathcal{P} be the current packing.

Now let B be a non-suitable block in \mathcal{P} , that is, $w(B) > c^3$ and no edge of its spanning tree T is oriented in both directions. This implies that there exists a vertex $v \in V(T)$ with all its incident edges pointing towards it. We call such a vertex v a *sink*. Let T_1, \dots, T_p be the connected components of $T \setminus \{v\}$. Note that as v is a sink, $w(T_i) < c$ for $1 \leq i \leq p$, using the fact that $w(B) > c^3$ we conclude that $p \geq c^2$. Now let P_1, \dots, P_ℓ be the connected components of $G[V(T_1) \cup \dots \cup V(T_p)] = G[V(B) \setminus \{v\}]$, and note that $\ell \leq p$. We call these subgraphs P_i the *pieces* of the non-suitable block B . For each non-suitable block, we delete the pieces with no white vertex. This completes the construction of \mathcal{P} . The next claim bounds the number of white vertices in each piece of a non-suitable block in \mathcal{P} .

Claim 1. *Each of the pieces of a non-suitable block contains less than c^2 white vertices.*

Proof. Assume for contradiction that there exists a piece P of a non-suitable block with $w(P) \geq c^2$, and let v be the sink of the non-suitable block obtained from tree T . By construction $V(P)$ is the union of the vertices in some of the trees in $T \setminus \{v\}$; let without loss of generality these trees be T_1, \dots, T_q . As $w(P) \geq c^2$ and $w(T_i) < c$ for $1 \leq i \leq q$, it follows that $q \geq c$. As v has at least one neighbor in each of the trees T_i , $1 \leq i \leq q$, and P is a connected subgraph of G , we can obtain a c -pumpkin-model $\{A, B\}$ in $G[V \setminus S]$ by setting $A := \{v\}$ and $B := V(P)$, contradicting the fact that $G[V \setminus S]$ is c -pumpkin-free. \square

Hence in the packing \mathcal{P} we are left with a set of suitable blocks with at most c^3 white vertices each, and a set of non-suitable blocks, each one broken up into pieces linked by a sink in a star-like structure. By Claim 1, each piece of the remaining non-suitable blocks contains at most c^2 white vertices.

Now we need two of claims about the properties of the constructed packing.

Claim 2. *In the packing \mathcal{P} constructed above, the number of suitable or non-suitable blocks is $\mathcal{O}(k)$.*

Proof. We first bound the number of suitable blocks and for this we distinguish between two types of suitable blocks:

- Type 1: blocks that have exactly one neighbor in S . For each such block we need to include a vertex of it in the c -hitting set (as each suitable block contains at least c white vertices), so obviously their number is at most k .
- Type 2: blocks that have at least two distinct neighbors in S . The number of such blocks is $\mathcal{O}(k)$ by Lemma 3.4.

The proof for non-suitable blocks is similar. We distinguish between the same two types of blocks, and use the fact that each non-suitable block contains at least $c^3 \geq c$ white vertices. This concludes the proof. \square

Claim 3. *In an (S, c) -reduced instance, either the total number of pieces in the packing \mathcal{P} constructed above is $\mathcal{O}(k)$ or we can branch according to branching rule **B**.*

Proof. We first distinguish between three types of pieces:

- Type 1: pieces that have at least two distinct neighbors in S . The number of pieces of this type is $\mathcal{O}(k)$ by Lemma 3.4.
- Type 2: pieces that are not of type 1 and that have at least one neighbor in some suitable block or in another non-suitable block (note that by construction a piece cannot have any neighbor in other pieces in the same non-suitable block). We construct an auxiliary graph H as follows: we start with the packing \mathcal{P} , and we add all the edges in $G[V \setminus S]$ between vertices in different blocks of \mathcal{P} (suitable or non-suitable). Then we contract each block to a single vertex, and let H be the resulting graph. By Claim 2, $|V(H)| = \mathcal{O}(k)$. As $H \preceq_m G[V \setminus S]$ and $G[V \setminus S]$ is c -pumpkin-free, by Corollary 2.4 we have that $|E(H)| = \mathcal{O}(k)$. The bound follows from the fact the number of pieces of Type 2 is at most $2|E(H)|$, as each piece is incident with at least one edge of H after uncontracting the blocks.
- Type 3: the remaining pieces. That is, these are pieces P that see exactly one vertex u in S , and that are connected to the rest of $G[V \setminus S]$ only through the corresponding sink v . In other words, such a piece P is a connected component of $G \setminus \{u, v\}$. Let

$$\alpha(P) := \max\{c' : G[V(P) \cup \{u\}] \text{ has a } c'\text{-pumpkin-minor}\}.$$

We distinguish two cases:

- If $\alpha(P) \geq c$, then any c -hitting set needs to contain at least one vertex in P . We conclude that the number of pieces of this subtype is at most k .
- Otherwise, $\alpha(P) < c$. Now we only focus on pieces of this subtype. We again distinguish two cases.

- Assume first that for some sink v , there are two pieces P_1 and P_2 in its block that have neighbors in distinct connected components of $G[S]$, say u and w respectively. If $|P_1 \cup P_2|$ is small, then we can branch according to branching rule **B**. Otherwise we apply protrusion rule **P** to reduce the size of the P_1 to $\mathcal{O}(1)$. Observe that P_1 contains at most $c' < c$ white vertices. As $\text{tw}(P_1 \cup \{v\}) \leq 2c$ and $|\partial(P_1 \cup \{v\})| \leq c' + 1$, $P_1 \cup \{v\}$ is a $2c$ -protrusion. Now we can replace (G, S, k) by another equivalent graph (G', S, k') ($k' \leq k$) such that the graph P_1 is replaced by P'_1 , it contains all the white vertices, and the size of P'_1 is $\mathcal{O}(1)$. Consider a connected components of P'_1 , say C . If $|\partial(C)| \leq 1$, then $\partial(C)$ is either v or u . In either case the conditions of reduction rule **R2** or **R3** hold and we can delete C . So either we can remove all connected components or there exists a connected component such that $\partial(C) = \{u, v\}$. In the first case we will get the following graph $(G \setminus P_1, S, k')$ and hence we have made progress. If the second case occurs then we do as follows.

Now we do the same with $P_2 \cup \{v\}$ as we did for $P_1 \cup \{v\}$ but on the initial input (G, S, k) instead of (G', S, k') . In this case either we can get rid of P_2 and obtain an equivalent instance or we have a connected component C' in P'_2 such that $\partial(C') = \{v, w\}$. So in the case we apply protrusion rule **P** on $P_1 \cup \{v\}$ and $P_2 \cup \{v\}$ and get two connected components C and C' with $\partial(C) = \{u, v\}$ and $\partial(C') = \{v, w\}$, respectively, we apply the reduction rules on both $P_1 \cup \{v\}$ and $P_2 \cup \{v\}$. Let (G^*, S, k^*) be the reduced instance and C and C' be the connected components as described above. Now the number of vertices in each connected component is $\mathcal{O}(1)$. Notice that there exists a path P in $G[C \cup C' \cup \{v\}]$ of constant size whose endpoints see different connected components of $G[S]$. Therefore, we can branch according to branching rule **B**.

If we do not branch then we repeat this step, in which case we either branch eventually or all the pieces associated with this sink have neighbors in the same connected component of $G[S]$.

- Otherwise, the pieces associated with each sink have neighbors in the same connected component of $G[S]$. We finally distinguish two more cases.
 - Assume that in some non-suitable block with sink v there are more than c pieces connected to the same connected component of $G[S]$. As $\alpha(P) < c$ and because we are dealing with pieces of Type 3, all the conditions of reduction rule **R4** are fulfilled and therefore vertex v can be safely included in the solution. Therefore, this case does not occur in an (S, c) -reduced instance.
 - Otherwise, the number of pieces of each non-suitable block is at most c . As each piece contains at most c^2 white vertices and the total number of non-suitable blocks is $\mathcal{O}(k)$ by Claim 2, the total number of white vertices in non-suitable blocks of this subtype is $\mathcal{O}(k)$.

So when we do not branch the number of pieces is $\mathcal{O}(k)$, concluding the proof. \square

Note that the proof of Claim 3 shows that if the number of pieces is not $\mathcal{O}(k)$, then we can apply protrusion rule **P** (and possibly reduction rules **R2** and **R3**) to eventually branch according to branching rule **B**.

To conclude, recall that the constructed packing \mathcal{P} contains all but $\mathcal{O}(k)$ white vertices, either in suitable blocks or in pieces of non-suitable blocks. As by construction each suitable block has at most c^3 white vertices and by Claim 2 the number of such blocks is $\mathcal{O}(k)$, it follows that the number of white vertices contained in suitable blocks is $\mathcal{O}(k)$. Similarly, by Claim 1 each piece contains at most c^2 white vertices, and the total number of pieces is $\mathcal{O}(k)$ by Claim 3 unless we can branch according to branching rule **B**, so if we cannot branch the number of white vertices contained in pieces of non-suitable blocks is also $\mathcal{O}(k)$. \square

Final algorithm. Finally we combine everything to obtain the following result.

Theorem 3.7. *For any fixed $c \geq 1$, the p -c-PUMPKIN-COVERING problem can be solved in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.*

Proof. To obtain the desired result, by Lemma 3.1 it is sufficient to obtain an algorithm for p -c-DISJOINT COVER. Recall the measure $\mu = \text{cc}(G[S]) + k$. Now by Lemma 3.6 either we branch according to branching rule **B**, or we can apply protrusion rule **P** and get a smaller instance, or we have that $|V_1| \leq h(c) \cdot k$. In the first case we branch into $\alpha(c)$ ways and in each branch the measure decreases by one as either we include a vertex in our potential solution or we decrease the number of connected components of $G[S]$. Here $\alpha(c)$ is a constant that only depends on c . This implies that the number of nodes in the branching tree is upper-bounded by $\alpha(c)^\mu \leq \alpha(c)^{2k+1} = 2^{\mathcal{O}(k)}$. In the second case, we get a smaller instance in polynomial time. Finally, in the third case, using Lemma 3.3 we get that $|A = V_1 \cup V_{\geq 2}| = \mathcal{O}(k)$. Thus using Lemma 3.2 we can get an equivalent instance (G^*, S, k^*) with $\mathcal{O}(k)$ vertices and hence the problem can be solved by enumerating all subsets of size at most k^* of $G^* \setminus S$. This concludes the proof. \square

4. AN APPROXIMATION ALGORITHM FOR COVERING AND PACKING PUMPKINS

In this section we show that every n -vertex graph G either contains a small c -pumpkin-model or has a structure that can be reduced, giving a smaller equivalent instance for both the MINIMUM c -PUMPKIN-COVERING and the MAXIMUM c -PUMPKIN-PACKING problems. Here by a “small” c -pumpkin-model, we mean a model of size at most $f(c) \cdot \log n$ for some function f independent of n . We finally use this result to derive a $\mathcal{O}(\log n)$ -approximation algorithm for both problems.

This section is organized as follows. We first describe in Section 4.1 our reduction rules and prove their validity for both covering and packing problems (see Lemma 4.1). The existence of small c -pumpkin-models in c -reduced graphs is provided in Section 4.3 (see Lemma 4.11); its proof strongly relies on a graph structure that we call *hedgheg*, which we study in Section 4.2. We finally focus in Section 4.4 on the algorithmic consequences of our results (see Theorems 4.12 and 4.13).

4.1. Reduction rules. We describe two reduction rules for hitting/packing c -pumpkin-models, which given an input graph G satisfying some specific conditions, produce a graph H with less vertices than G and satisfying $\tau_c(G) = \tau_c(H)$ and $\nu_c(G) = \nu_c(H)$. Moreover, these operations are defined in such a way that, for both problems, an optimal (approximate) solution for G can be retrieved in polynomial time from an optimal (resp. approximate) solution for H .

An *outgrowth* of a graph G is a triple (C, u, v) such that



FIGURE 2. The graph $\Gamma(C, u, v)$ of two outgrowths (C, u, v) . We have $\lambda(C, u, v) = 8 < 9 = \gamma(C, u, v)$ in the left one, while $\lambda(C, u, v) = 7 > 5 = \gamma(C, u, v)$ holds for the right one.

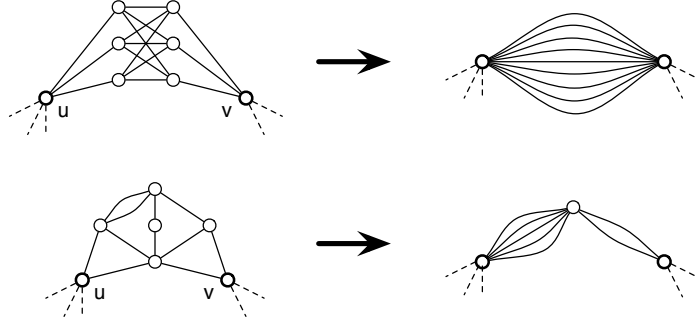


FIGURE 3. Illustration of reduction rule **Z2** on the two outgrowths from Fig. 2.

- (i) u, v are two distinct vertices of G ;
- (ii) C is a connected component of $G \setminus \{u, v\}$ with $|C| \geq 2$; and
- (iii) u and v both have at least one neighbor in C in the graph G .

Given an outgrowth (C, u, v) of a graph G , we let $\Gamma(C, u, v)$ denote the subgraph of G induced by $V(C) \cup \{u, v\}$ where all the edges between u and v are removed. We let $\Lambda(C, u, v)$ be the graph $\Gamma(C, u, v)$ where an edge between u and v has been added. Also, we define $\gamma(C, u, v)$ as the largest integer k such that $\Gamma(C, u, v)$ has a k -pumpkin-model $\{A, B\}$ with $u \in A$ and $v \in B$, and $\lambda(C, u, v)$ as the largest integer k such that $\Lambda(C, u, v)$ has a k -pumpkin-model $\{A, B\}$ with $u, v \in A$. See Fig. 2 for an illustration.

Now that we are equipped with these definitions and notations, we may describe the two reduction rules, which depend on the value of the positive integer c .

- Z1** Suppose v is a vertex of G such that no block of G containing v has a c -pumpkin-minor. Then define H as the graph obtained from G by removing v .
- Z2** Suppose (C, u, v) is an outgrowth of G such that $\Gamma(C, u, v)$ has no c -pumpkin-minor. Assume further that **Z1** cannot be applied on G . Let $\gamma := \gamma(C, u, v)$ and $\lambda := \lambda(C, u, v)$. If $\lambda \leq \gamma$, define H as the graph obtained from $G \setminus V(C)$ by adding γ parallel edges between u and v . Otherwise, define H as the graph obtained from $G \setminus V(C)$ by adding a new vertex v_C and linking v_C to u with γ parallel edges, and to v with $\lambda - \gamma$ parallel edges.

See Fig. 3 for an illustration of **Z2**. We note that testing for the existence of a k -pumpkin-model $\{A, B\}$ in a graph G can be done in polynomial time when k is a fixed constant, and this remains true if we moreover require $u \in A$ and/or $v \in B$, where u, v are two specified vertices of G . This follows from classical results of Robertson and Seymour [26]. In particular, the parameters $\gamma(C, u, v)$ and $\lambda(C, u, v)$ can be computed

in polynomial time when C has no c -pumpkin minor¹. Therefore, the reduction rule **Z2** can be applied in polynomial time.

A graph G is said to be c -reduced if neither **Z1** nor **Z2** can be applied to G . The next lemma shows the validity of these reduction rules.

Lemma 4.1. *Let c be a fixed positive integer. Suppose that H results from the application of **Z1** or **Z2** on a graph G . Then*

- (a) $\tau_c(G) = \tau_c(H)$ and moreover, given a c -hitting set X' of H , one can compute in polynomial time a c -hitting set X of G with $|X| \leq |X'|$.
- (b) $\nu_c(G) = \nu_c(H)$ and moreover, given a c -packing \mathcal{M}' of H , one can compute in polynomial time a c -packing \mathcal{M} of G with $|\mathcal{M}| = |\mathcal{M}'|$.

In order to prove Lemma 4.1, we first need to introduce a few technical lemmas; the validity of the reduction rules is shown in Lemmas 4.5 and 4.6 at the end of this section, which correspond to Lemma 4.1(a) and Lemma 4.1(b), respectively.

Lemma 4.2. *Let (C, u, v) be an outgrowth of a graph G , and suppose that **Z1** cannot be applied on G . Then $\lambda(C, u, v) \leq 2\gamma(C, u, v)$.*

Proof. Let $\gamma := \gamma(C, u, v)$, $\Gamma := \Gamma(C, u, v)$, $\lambda := \lambda(C, u, v)$, and $\Lambda := \Lambda(C, u, v)$. Let $\{A, B\}$ be a λ -pumpkin-model in Λ with $u, v \in A$, and such that B is inclusion-wise maximal with that property. Observe that if there is some vertex x in $V(\Lambda) \setminus A$ that is adjacent to some vertex in B in the graph Λ , then we could have added x to B , which contradicts its maximality. Hence, every vertex outside B that is adjacent to some vertex in B is included in A . We will use this fact later in the proof.

If Λ is not 2-connected, then Λ has a cutvertex x (since $|\Lambda| \geq 3$). The vertex x cannot be u nor v , because $\Lambda - u$ and $\Lambda - v$ are both connected. Also, the two vertices u, v are in the same block of Λ (because of the edge uv). It follows that there is a block K of Λ that includes x but avoids u and v . Then K is also a block of G , and we could have applied **Z1** on any vertex in $V(K) \setminus \{x\}$, a contradiction. Therefore, Λ must be 2-connected.

Let us call a vertex in $V(\Lambda) \setminus B$ that is adjacent to at least one vertex in B simply a *neighbor of B* . Since Λ is 2-connected, B has at least two neighbors. Also, as we have seen previously, every neighbor of B is in A .

For every two distinct neighbors x, y of B , there are two vertex-disjoint paths P_1, P_2 between $\{u, v\}$ and $\{x, y\}$ in Λ , since the graph is 2-connected. Let us choose such vertices x, y and paths P_1, P_2 in such a way that $|P_1| + |P_2|$ is minimized. Exchanging P_1 with P_2 if necessary, we may assume $u \in V(P_1)$ and $v \in V(P_2)$. It follows from our choice of x, y, P_1, P_2 that $V(P_1) \cap B = \emptyset$ and $V(P_2) \cap B = \emptyset$.

The paths P_1 and P_2 also exist in the graph Γ , since they cannot use the edge uv . Let us build two disjoint sets S_u and S_v of vertices iteratively as follows. First let $S_u := V(P_1)$ and $S_v := V(P_2)$. Then, for every neighbor z of B that is distinct from x and y , consider a shortest path Q between $S_u \cup S_v$ and $\{z\}$ in $\Gamma[A]$. (Such a path exists since $z \in A$, and $\Gamma[A]$ is either connected, or has two connected components and u, v are in distinct components.) Then $V(Q)$ has a non-empty intersection with either S_u or S_v (but not both); add all vertices of $V(Q)$ to S_u in the first case, to S_v in the second case.

By definition, S_u, S_v , and B are mutually disjoint, the two graphs $\Gamma[S_u]$ and $\Gamma[S_v]$ are connected, and every neighbor of B is in either S_u or S_v . Thus, letting p and q be the number of S_u - B edges and S_v - B edges in Γ , respectively, we have $p + q = \lambda$. On the

¹ We note that $\lambda(C, u, v)$ can equivalently be defined as the largest integer k such that $\Lambda(C, u, v)/uv$ has a k -pumpkin-model $\{A, B\}$ with $u \in A$.

other hand, $\{S_u, B \cup S_v\}$ is a p -pumpkin-model in Γ with $u \in S_u$, $v \in B \cup S_v$ implying $\gamma \geq p$. Similarly, $\{S_u \cup B, S_v\}$ is a q -pumpkin-model in Γ with $u \in S_u \cup B$, $v \in S_v$, showing $\gamma \geq q$. Therefore, $\lambda = p + q \leq 2\gamma$, as claimed. \square

Lemma 4.3. *Let c be a fixed positive integer. Suppose H is obtained by applying rule **Z2** on an outgrowth (C, u, v) of a graph G . Let X be an arbitrary subset of vertices of $V(G) \setminus (V(C) \cup \{u, v\})$. Then, given a c -pumpkin-model of $H \setminus X$, one can find in polynomial time a c -pumpkin-model of $G \setminus X$.*

Proof. Let $\Gamma := \Gamma(C, u, v)$, $\Lambda := \Lambda(C, u, v)$, $\gamma := \gamma(C, u, v)$ and $\lambda := \lambda(C, u, v)$. Let $\{A, B\}$ denote the given c -pumpkin-model of $H \setminus X$. We may assume that this model is minimal (if not, one can obviously make it minimal in polynomial time).

Case 1: $\lambda \leq \gamma$. If $u \notin A \cup B$ or $v \notin A \cup B$, then $\{A, B\}$ is a c -pumpkin-model in $G \setminus X$ and we are done. Thus, exchanging A and B if necessary, we may assume that either $u, v \in A$, or $u \in A$ and $v \in B$. In the first case, since $G[A \cup V(C)]$ is connected, $\{A \cup V(C), B\}$ is a c -pumpkin-model in $G \setminus X$. In the second case, A and B both induce connected subgraphs of G , and there are γ extra edges between A and B in H compared to G . Let $\{A', B'\}$ be a γ -pumpkin-model in Γ with $u \in A'$ and $v \in B'$. Then $\{A \cup A', B \cup B'\}$ is a c -pumpkin-model in $G \setminus X$, as desired.

Case 2: $\lambda \geq \gamma + 1$. If $u \notin A \cup B$ or $v \notin A \cup B$, then by minimality $\{A, B\}$ also avoids the vertex v_C , and $\{A, B\}$ is a c -pumpkin-model in $G \setminus X$. Thus again, exchanging A and B if necessary, we may assume that either $u, v \in A$, or $u \in A$ and $v \in B$. Suppose the former holds. If $v_C \notin B$ then setting $A' := (A \setminus \{v_C\}) \cup V(C)$, we deduce that $\{A', B\}$ is a c -pumpkin-model in $G \setminus X$. (Note that this is true independently of whether $v_C \in A$ or not.) If $v_C \in B$, then $B = \{v_C\}$ and the number of edges between A and B in H is $\gamma + (\lambda - \gamma) = \lambda$. Hence, $\lambda \geq c$. Letting $\{A', B'\}$ be a λ -pumpkin-model in Λ with $u, v \in A'$, we have that $G[A \cup A']$ is connected. It follows that $\{A \cup A', B'\}$ is a c -pumpkin-model in $G \setminus X$.

Now assume $u \in A$ and $v \in B$. By Lemma 4.2 we have $\lambda \leq 2\gamma$, and thus $\lambda - \gamma \leq \gamma$. If $v_C \notin A \cup B$, then $\{A, B\}$ is a c -pumpkin-model of $G \setminus X$. If $v_C \in A \cup B$, then the number of edges in H between A and B that are incident with v_C is at most $\max\{\gamma, \lambda - \gamma\} = \gamma$. Hence, letting $\{A', B'\}$ be a γ -pumpkin-model of Γ with $u \in A'$ and $v \in B'$, we deduce that $\{A \cup A', B \cup B'\}$ is a c -pumpkin-model in $G \setminus X$.

Finally, we note that each step of the above proof can be realized in polynomial time. This follows from the fact that the parameters γ and λ and the associated models can be computed in polynomial time. \square

Next we show that the converse of the above lemma also holds.

Lemma 4.4. *Let $c, G, H, (C, u, v)$, and X be as in Lemma 4.3. Then, given a c -pumpkin-model of $G \setminus X$, one can find in polynomial time a c -pumpkin-model of $H \setminus X$.*

Proof. Let $\Gamma := \Gamma(C, u, v)$, $\Lambda := \Lambda(C, u, v)$, $\gamma := \gamma(C, u, v)$ and $\lambda := \lambda(C, u, v)$. Let $\{A, B\}$ denote the given c -pumpkin-model of $G \setminus X$. We may assume that this model is minimal.

Case 1: $\lambda \leq \gamma$. If $u \notin A \cup B$ or $v \notin A \cup B$, then $A, B \cap V(C) = \emptyset$ (by minimality of $\{A, B\}$) and thus $\{A, B\}$ is a c -pumpkin-model in $H \setminus X$. Thus, exchanging A and B if necessary, we may assume that either $u, v \in A$, or $u \in A$ and $v \in B$. First suppose the former holds. If B has no vertex in C , then $\{A \setminus V(C), B\}$ is a c -pumpkin-model in $H \setminus X$. If B contains some vertex of C then $B \subseteq V(C)$. Since $\{A \cap V(\Lambda), B\}$ is a c -pumpkin-model in Λ with $u, v \in A \cap V(\Lambda)$, we have $\lambda \geq c$. On the other hand, we have $\gamma < c$, since Γ has no c -pumpkin-minor. It follows that $c \leq \lambda \leq \gamma < c$, a contradiction. Thus B cannot contain a vertex of C , which concludes the case where $u, v \in A$.

Now assume $u \in A$ and $v \in B$. In the graph Γ , the number of edges between $A \cap V(\Gamma)$ and $B \cap V(\Gamma)$ is at most γ (by definition of γ). Since there are γ extra uv edges in H compared to G , we deduce that $\{A \setminus V(C), B \setminus V(C)\}$ is a c -pumpkin-model in $H \setminus X$, as desired.

Case 2: $\lambda \geq \gamma + 1$. For the same reasons as in the $\lambda \leq \gamma$ case, we must have $u, v \in A \cup B$, and thus we may assume that either $u, v \in A$, or $u \in A$ and $v \in B$. Suppose the former holds. If B does contain any vertex in C then $\{(A \setminus V(C)) \cup \{v_C\}, B\}$ is a c -pumpkin-model of $H \setminus X$. If B contains some vertex in C then $B \subseteq V(C)$, and $\{A \cap V(\Lambda), B\}$ is a c -pumpkin-model of Λ with $u, v \in A \cap V(\Lambda)$, implying $\lambda \geq c$, and that $\{A \setminus V(C), \{v_C\}\}$ is a c -pumpkin-model of $H \setminus X$.

Now assume $u \in A$ and $v \in B$. In Γ , there are at most γ edges between $A \cap V(\Gamma)$ and $B \cap V(\Gamma)$. Thus $\{A \setminus V(C), (B \setminus V(C)) \cup \{v_C\}\}$ is a c -pumpkin-model in $H \setminus X$. \square

Lemma 4.5. *Let c be a fixed positive integer. Suppose H results from the application of **Z1** or **Z2** on a graph G . Then $\tau_c(G) = \tau_c(H)$. Moreover, given a c -hitting set X' of H one can compute in polynomial time a c -hitting set X of G with $|X| \leq |X'|$.*

This lemma implies that an optimal solution to the MINIMUM c -PUMPKIN-COVERING problem on G can be computed given one for H , and similarly that an approximate solution for G can be obtained from an approximate solution for H . This will be used in our approximation algorithms in Section 4.4.

Proof of Lemma 4.5. First suppose H results from the application of **Z1** on G with vertex v . We trivially have $\tau_c(G) \geq \tau_c(H)$. Let X' be a given c -hitting set of H . If X' is not a c -hitting set of G , then $G \setminus X'$ has a c -pumpkin-model; let $\{A, B\}$ be a minimal one. We have $v \in A \cup B$ since otherwise $\{A, B\}$ would be a c -pumpkin-model in $H \setminus X'$. By the minimality of $\{A, B\}$, we must have $A \cup B \subseteq V(K)$ for some block K of G . But then K is a block of G including v and containing a c -pumpkin-minor, contradicting the assumptions of **Z1**. Therefore $X := X'$ is a c -hitting set of G , and $\tau_c(G) \leq \tau_c(H)$ also holds, implying $\tau_c(G) = \tau_c(H)$.

Now assume H has been obtained by applying **Z2** on G with outgrowth (C, u, v) . Let $\Gamma := \Gamma(C, u, v)$, $\Lambda := \Lambda(C, u, v)$, $\gamma := \gamma(C, u, v)$ and $\lambda := \lambda(C, u, v)$.

Observe that every minimal c -pumpkin-model of G containing a vertex of C must contain both u and v . This is because Γ has no c -pumpkin-minor. We also note that, in order to cover all c -pumpkin-models of G , it is enough to cover those that are minimal.

First we show $\tau_c(G) \geq \tau_c(H)$. Let X be a minimum c -hitting set of G . If $u \in X$ or $v \in X$, then X is trivially a c -hitting set of H , so let us assume $u, v \notin X$. Moreover, we may suppose that X has no vertex in C , since otherwise we could replace all such vertices with the vertex u (or equivalently v). Since $X \subseteq V(G) \setminus (V(C) \cup \{u, v\})$ and $G \setminus X$ has no c -pumpkin-minor, it follows from Lemma 4.3 that $H \setminus X$ has no c -pumpkin-minor either, that is, X is also a c -hitting set of H . This shows $\tau_c(G) \geq \tau_c(H)$.

Now we prove that $\tau_c(G) \leq \tau_c(H)$ also holds. Here we show that, given a c -hitting set X' of H , one can find in polynomial time a c -hitting set X of G with $|X| \leq |X'|$. Hence, this will also prove the second part of the lemma.

First, if $\lambda > \gamma$ and $v_C \in X'$, we replace v_C in X' by one of u, v ; the modified set X' clearly remains a c -hitting set of H , and its size did not increase (note that it could decrease if u and v were already in X'). Thus we may assume $X' \subseteq V(G) \setminus (V(C) \cup \{u, v\})$.

If $u \in X'$ or $v \in X'$, then $X := X'$ is also a c -hitting set of G , since Γ has no c -pumpkin-minor. If $u, v \notin X'$, then Lemma 4.4 implies that $G \setminus X'$ has no c -pumpkin-minor. Hence, $X := X'$ is a c -hitting set of G . This shows $\tau_c(G) \geq \tau_c(H)$, and therefore $\tau_c(G) = \tau_c(H)$. \square

We conclude this section with a lemma similar to Lemma 4.5 for c -packings.

Lemma 4.6. *Let c be a fixed positive integer. Suppose H results from the application of **Z1** or **Z2** on a graph G . Then $\nu_c(G) = \nu_c(H)$. Moreover, given a c -packing \mathcal{M}' of H one can compute in polynomial time a c -packing \mathcal{M} of G with $|\mathcal{M}| = |\mathcal{M}'|$.*

Proof. First suppose H results from the application of **Z1** on G with vertex v . Clearly, every c -packing of H is a c -packing for G . Thus $\nu_c(G) \geq \nu_c(H)$, and it is enough to show the reverse inequality. Consider a c -packing of G . We may assume that every c -pumpkin-model in that packing is minimal. Thus each such model is contained in some block of G , and hence avoids the vertex v . Therefore the packing also exists in H , implying $\nu_c(G) \leq \nu_c(H)$ and $\nu_c(G) = \nu_c(H)$, as desired.

Now assume H has been obtained by applying **Z2** on G with outgrowth (C, u, v) . Let $\gamma := \gamma(C, u, v)$ and $\lambda := \lambda(C, u, v)$.

First we show $\nu_c(G) \geq \nu_c(H)$. Let $\mathcal{M}' = \{M'_1, \dots, M'_k\}$ be a given c -packing of H . We show that a packing of the same size in G can be computed in polynomial time, which will prove the second part of the lemma. We may assume that each M'_i is minimal (if not, they can be made minimal in polynomial time). Thus if $\lambda > \gamma$ and M'_i contains the vertex v_C then M'_i also includes both u and v . It follows that if every M'_i avoids at least one of u, v then the packing $\mathcal{M} := \mathcal{M}'$ is a c -packing in G and we are done. So assume one model in the collection, say without loss of generality M'_1 , includes both u and v . Let X be the union of the vertices in M'_2, \dots, M'_k . Since M'_1 is a c -pumpkin-model in $H \setminus X$, using Lemma 4.3 we can compute in polynomial time a c -pumpkin-model M_1 in $G \setminus X$. Hence $\mathcal{M} := \{M_1, M'_2, \dots, M'_k\}$ is a c -packing of the desired size in G .

In order to prove $\nu_c(G) = \nu_c(H)$ it remains to show $\nu_c(G) \leq \nu_c(H)$. Let $\{M_1, \dots, M_k\}$ be a c -packing of G . Again we may assume that each M_i is minimal. Thus if some M_i contains some vertex of C then M_i contains both u and v . If there is no such model in the packing then $\{M_1, \dots, M_k\}$ is also a c -packing of H and we are done. We may thus assume that some model in the packing, say without loss of generality M_1 , contains both u and v . As before, let X be the union of the vertices in M_2, \dots, M_k . Using Lemma 4.4 with M_1 we find a c -pumpkin-model M'_1 in $H \setminus X$. Thus $\{M'_1, M_2, \dots, M_k\}$ is a c -packing of size k in H , as desired. \square

4.2. Hedgehogs. A graph is said to be a *multipath* if its underlying simple graph is isomorphic to a path. If P is a multipath and $u, v \in V(P)$, we write uPv for the subgraph of P induced by the vertices on a u - v path in P (thus edges in uPv have the same multiplicities as in P).

A *hedgehog* is a pair (H, P) , where H is a graph and P is an induced multipath of H with $|P| \geq 2$ and such that

- (i) the (possibly empty) set $S := V(H) \setminus V(P)$ is a stable set of H ; and
- (ii) every vertex in S has at least two neighbors in P .

(Let us recall that a *stable set* is a set of vertices such that no two of them are adjacent.)

Consider a hedgehog (H, P) . Its *size* is defined as $|P|$, the number of vertices in P . A *bad cutset* of (H, P) is a set $X = \{u, v\}$ of two *internal* vertices of P such that $H \setminus X$ has a connected component C avoiding both endpoints of P with $|C| \geq 2$. This definition is motivated by reduction rule **Z2**: if C is such a component, then (C, u, v) is an outgrowth of H . A *rooted c -pumpkin-model* of (H, P) is a c -pumpkin-model $\{A, B\}$ of H with the extra property that A and B both contain an endpoint of P .

Given a hedgehog (H, P) and a connected induced subgraph Q of P with $|Q| \geq 2$, one can define a hedgehog (H', Q) as follows: First, remove from H every vertex not in P that has no neighbor in Q . Then contract every edge of P not included in Q . Finally, remove from the graph every vertex not in Q that has only one neighbor in Q .

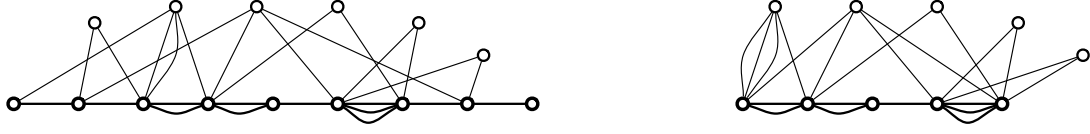


FIGURE 4. A hedgehog (H, P) (left) and a contraction (H', Q) of (H, P) (right). The multipaths P and Q are drawn in bold.

This defines the graph H' . We leave it to the reader to check that (H', Q) is indeed a hedgehog; we say that (H', Q) is the *contraction* of (H, P) on the multipath Q . See Fig. 4 for an illustration of this operation. The following lemma is a direct consequence of the definition.

Lemma 4.7. *If (H', Q) is a contraction of a hedgehog (H, P) and X is a bad cutset of (H', Q) , then X is also a bad cutset of (H, P) .*

We show that every big enough hedgehog has a rooted c -pumpkin-model or a bad cutset. This fact will be useful in the subsequent proofs.

Lemma 4.8. *Let c be a fixed positive integer. Then every hedgehog (H, P) of size at least $(4c)^{4c}$ contains a rooted c -pumpkin-model or a bad cutset, either of which can be found in polynomial time.*

Proof. The proof is by induction on c . The base case $c = 1$ is trivial since P directly gives a rooted 1-pumpkin-model. For the inductive step, assume $c > 1$. Define $f(k)$, for a positive integer k , as $f(k) := (4k)^{4k}$. Let $S := V(H) \setminus V(P)$. Let a, b be the endpoints of P .

If a vertex $v \in S$ has at least c neighbors in P , then let w be the neighbor of v that is closest to a on P . Then $A := V(aPw) \cup \{v\}$ and $B := V(P) \setminus A$ both induce a connected subgraph of H . Moreover, there are at least $c - 1$ edges from v to B , and at least one from $A \setminus \{v\}$ to B (because of P). Since $a \in A$ and $b \in B$, we deduce that $\{A, B\}$ is a rooted c -pumpkin-model of (H, P) . Thus we may assume that every vertex in S has at most $c - 1$ neighbors in P . In particular we have $c \geq 3$, since every vertex in S has at least two neighbors in P .

The multipath P , seen from its endpoint a , induces a natural linear ordering of the neighbors of a given vertex in S ; we say that two such neighbors are *consecutive* if they are consecutive in that ordering.

Suppose that there exists a vertex $v \in S$ with two consecutive neighbors x, y such that $|xPy| \geq f(c - 1) + 2$. Consider the contraction (H', Q) of (H, P) on the multipath $Q := xPy \setminus \{x, y\}$. Since $|Q| \geq f(c - 1)$, by induction (H', Q) has a rooted $(c - 1)$ -pumpkin-model $\{A', B'\}$ or a bad cutset X . If the latter holds, then by Lemma 4.7 the set X is also a bad cutset of (H, P) and we are done. Thus we may assume the former holds. In the graph H , the vertex v has no neighbor in Q , thus v is not included in H' . Hence, we can obtain a rooted c -pumpkin-model $\{A, B\}$ in (H, P) by setting $A := A' \cup V(aPx) \cup \{v\}$ and $B := B' \cup V(yPb)$. Therefore we can assume that, for every vertex $v \in S$, every two consecutive neighbors of v are at distance at most $f(c - 1)$ on P .

Let us enumerate the vertices of P in order as p_1, p_2, \dots, p_k , with $p_1 = a$ and $p_k = b$. We may assume that, for every $i \in \{2, \dots, k - 4\}$,

$$(2) \quad \text{at least one of } p_{i+1}, p_{i+2} \text{ is adjacent to some vertex in } S.$$

Indeed, if not then $\{p_i, p_{i+3}\}$ would be a bad cutset of (H, P) . Since $k = |P| \geq f(c) \geq f(3) > 5$, this implies in particular that S is not empty.

Define an *open* interval $I_v = (i, j)$ for every vertex $v \in S$, where i (j) is the smallest (largest, respectively) index t such that p_t is a neighbor of v in H . (Observe that $i < j$ since v has at least two neighbors.) Now, let G be the interval graph defined by these open intervals, that is, let $V(G) := S$, and for every two distinct vertices $v, w \in S$, make v adjacent to w in G if and only if $I_v \cap I_w \neq \emptyset$.

For a connected subgraph G' of G , we define $I(G')$ as the union of the intervals of vertices in G' , that is, $I(G') := \bigcup \{I_v : v \in V(G')\}$. Observe that, since G' is connected, we have $I(G') = (i, j)$ for some integers i, j with $1 \leq i < j \leq k$.

First suppose that G has at least five connected components. The ordering p_1, \dots, p_k of the vertices of P induces an ordering of these components; let C, C', C'' be three consecutive connected components in that ordering, with the property that C is not first and C'' is not last in the ordering. Let $(i, j) := I(C)$, $(i', j') := I(C')$, and $(i'', j'') := I(C'')$. Then we have $1 < i < j \leq i' < j' \leq i'' < j'' < k$, and every vertex of S that is adjacent to some vertex in $\{p_{i+1}, \dots, p_{j''-1}\}$ has all its neighbors in that set. Since $j'' \geq i + 3$, it follows that $\{p_i, p_{j''}\}$ is a bad cutset of (H, P) . Hence, we may assume that G has at most four connected components.

A vertex p_t ($t \in \{1, \dots, k\}$) is said to be *exposed* if there is no connected component C of G with interval $I(C) = (i, j)$ such that $i \leq t \leq j$. If G has q connected components, it follows from (2) that at most $q + 5$ vertices of P are exposed. Since $q \leq 4$, it follows that G has a connected component C such that $I(C) = (x, y)$ with

$$(3) \quad y - x + 1 \geq \frac{|P| - 9}{4} \geq \frac{f(c) - 9}{4}.$$

Let $Q := p_x P p_y$ and let (H', Q) be the contraction of (H, P) on Q . We will show that (H', Q) contains a rooted c -pumpkin-model. The lemma will then follow, since such a model can be extended straightforwardly to one of (H, P) .

First let us observe that H' is an induced subgraph of H (by our choice of Q). Let $S' := V(H') \setminus V(Q) = V(C)$. For a vertex $v \in S'$, let us denote by $l(v)$ and $r(v)$ the two integers such that $I_v = (l(v), r(v))$.

It follows from our assumptions on (H, P) that, for every vertex $v \in S'$, the vertex v has at most $c - 1$ neighbors in Q and every two consecutive neighbors of v are at distance at most $f(c - 1)$ on Q . This implies

$$(4) \quad r(v) - l(v) \leq (c - 2)f(c - 1)$$

for every $v \in S'$.

In H' , the vertices p_x and p_y each have at least one neighbor in S' . Let $v \in S'$ be a neighbor of p_x maximizing $r(v)$, and let $w \in S'$ be a neighbor of p_y minimizing $l(w)$. Let Z be a shortest v - w path in the interval graph C ; enumerate the vertices of Z as z_1, z_2, \dots, z_m with $z_1 = v$ and $z_m = w$.

By our choice of v, w and the fact that Z is a shortest v - w path, we have

$$(5) \quad l(z_j) < l(z_{j+1})$$

$$(6) \quad r(z_j) < r(z_{j+1})$$

for every $j \in \{1, \dots, m - 1\}$, and

$$(7) \quad r(z_j) \leq l(z_{j+2})$$

for every $j \in \{1, \dots, m - 2\}$.

Since the interval $I(Z) = I(C) = (x, y)$, it follows from (3) and (4) that

$$(8) \quad m \geq \frac{y - x}{(c - 2)f(c - 1)} \geq \frac{f(c) - 13}{4(c - 2)f(c - 1)} = \frac{(4c)^{4c} - 13}{4(c - 2)(4c - 4)^{4c-4}} > c.$$

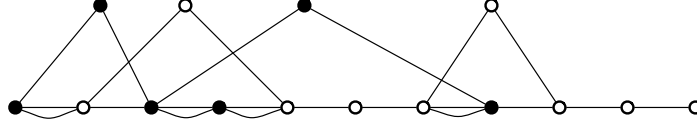


FIGURE 5. Illustration of the coloring for $d = 2$. The vertices in $\{z_1, \dots, z_{2d}\}$ are on top, the path Q at the bottom, and each breakpoint p_i of Q is linked to the vertices z_j such that $j \in J(p_i)$.

Let $d := \lfloor m/2 \rfloor$. Define, for $i \in \{x, \dots, y\}$, the set $J(p_i)$ as the set of indices $j \in \{1, \dots, 2d\}$ such that $i \in \{l(z_j), r(z_j)\}$. We say that p_i is a *breakpoint* of Q if $J(p_i)$ is not empty. (Thus p_x is a breakpoint in particular.) It is a consequence of (5), (6), and (7) that, if $|J(p_i)| > 1$, then $J(p_i) = \{j, j+2\}$ for some $j \in \{1, \dots, 2d-2\}$. In particular, the numbers in $J(p_i)$ always have the same parity.

We color the vertices in $V(Q) \cup \{z_1, \dots, z_{2d}\}$ in *black* or *white* as follows. First, for every $j \in \{1, \dots, 2d\}$, color z_j black if j is odd, white if j is even. Next color every breakpoint p_i of Q with the color corresponding to the parity of the numbers in $J(p_i)$ (namely, black for odd and white for even). Finally, color every uncolored vertex of Q with the color of the closest breakpoint of Q in the direction of p_x . See Fig. 5 for an illustration of the coloring.

Let A and B be the set of black and white vertices, respectively. By construction, $p_x \in A$ and $p_y \in B$, and each of A, B induces a connected subgraph of H' . Moreover, there are $2d+1 \geq m \geq c$ edges of Q whose endpoints received distinct colors. It follows that $\{A, B\}$ is a rooted c -pumpkin-model of (H', Q) , as desired.

We have shown that (H, P) always has a rooted c -pumpkin-model or a bad cutset. Moreover, it is easily seen from the proof given above that each of these can be found in polynomial time. This concludes the proof of the lemma. \square

We note that no effort has been made in order to optimize the constants in Lemma 4.8.

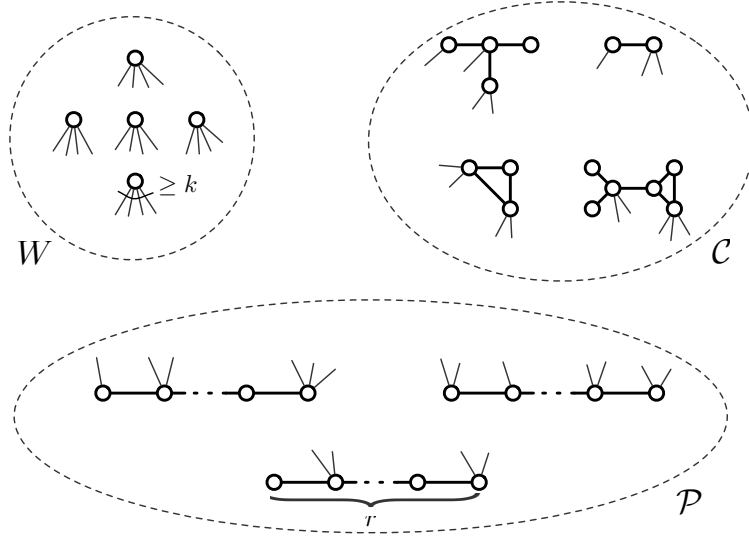
4.3. Small pumpkins in reduced graphs. Our goal is to prove that every n -vertex c -reduced graph G has a c -pumpkin-model of size $\mathcal{O}_c(\log n)$. We will use the following recent result by Fiorini *et al.* [16] about the existence of small minors in *simple* graphs with large average degree.

Theorem 4.9 (Fiorini *et al.* [16]). *There is a function h such that every n -vertex simple graph G with average degree at least 2^t contains a K_t -model with at most $h(t) \cdot \log n$ vertices.*

Even though the authors of [16] do not mention it explicitly, we note that the proof given in that paper can easily be turned into a poly-time algorithm finding such a K_t -model. Since a K_t -model in a graph directly gives a c -pumpkin-model of the same size for $c = (\lfloor t/2 \rfloor)^2$, we have the following corollary from Theorem 4.9, which is central in the proof of Lemma 4.11.

Corollary 4.10. *There is a function h such that every n -vertex simple graph G with average degree at least $2^{2\sqrt{c}+1}$ contains a c -pumpkin-model with at most $h(c) \cdot \log n$ vertices. Moreover, such a model can be computed in polynomial time.*

The next lemma states the existence of small c -pumpkin-models in c -reduced graphs; its proof strongly relies on the graph structure that we have called *hedgehog* (see Section 4.2). The idea is that an *hedgehog* witnesses the existence of either a small c -pumpkin-model or an outgrowth, which is impossible in a c -reduced graph.

FIGURE 6. The sets W , \mathcal{P} , and \mathcal{C} in the graph G .

Lemma 4.11. *There is a function f such that every n -vertex c -reduced graph G contains a c -pumpkin-model of size at most $f(c) \cdot \log n$. Moreover, such a model can be computed in polynomial time.*

Proof. Let

$$\begin{aligned} k &:= c^2 \left\lceil 2^{2\sqrt{c}+1} \right\rceil \\ r &:= (4c)^{4c} k \\ b &:= k^r. \end{aligned}$$

We will prove the lemma with f defined as

$$f(c) := \max\{krb, 3rc \cdot h(c)\},$$

where h is the function in Corollary 4.10. Throughout the proof, a c -pumpkin-model of G is said to be *small* if it has the required size, that is, if it has at most $f(c) \log n$ vertices.

The lemma trivially holds if $\mu(G) \geq c$, so we may assume $\mu(G) < c$. Let W be the (possibly empty) subset of vertices of G having degree at least k .

We build a collection \mathcal{P} of vertex-disjoint induced subgraphs of $G \setminus W$, each isomorphic to a multipath on r vertices. Initially, we let $\mathcal{P} := \emptyset$ and $G' := G \setminus W$. Then, as long as G' has a connected component C with diameter at least $r - 1$, we do the following: First, we consider two vertices at distance $r - 1$ in C and compute a shortest path Q between these two vertices. Note that the subgraph P of G induced by $V(Q)$ is a multipath on r vertices. Next, we add P to \mathcal{P} . Finally, we remove from G' the r vertices in P .

When the above procedure is finished, every connected component of G' has diameter less than $r - 1$ and maximum degree less than k . Hence each such component has bounded size: at most $k^r = b$ vertices. Let \mathcal{C} denote the collection of connected components of G' .

An illustration of the sets W , \mathcal{P} , and \mathcal{C} in the graph G is given in Fig. 6.

If some subgraph $C \in \mathcal{C}$ contains a c -pumpkin-model, then the size of the model is at most $|C| \leq b \leq f(c) \leq f(c) \log n$, and we are done. Thus we may assume that no subgraph $C \in \mathcal{C}$ contains a c -pumpkin-minor.

Let J be the graph obtained from G by contracting each subgraph $C \in \mathcal{C}$ into a single vertex v_C . Consider a subgraph $C \in \mathcal{C}$. We cannot have $\deg_J^*(v_C) = 0$, because otherwise we could have applied **Z1** on any vertex v of C in G (since C has no c -pumpkin-minor). If $\deg_J^*(v_C) = 1$, then let v be an arbitrary vertex of C , and let w be the unique vertex in $V(G) \setminus V(C)$ having a neighbor in $V(C)$ in the graph G . Since **Z1** cannot be applied on G with vertex v , there is a block B of G that includes v and containing a c -pumpkin-model. Since $V(B) \subseteq V(C) \cup \{w\}$, this model has size at most $|B| \leq b + 1 \leq f(c)$, that is, we have found a small c -pumpkin-model of G . Therefore, we may assume

$$(9) \quad \deg_J^*(v_C) \geq 2$$

for every $C \in \mathcal{C}$.

Let K be the graph obtained from J by contracting each subgraph $P \in \mathcal{P}$ into a single vertex v_P . If two vertices of K are linked by at least c parallel edges, then we directly find a c -pumpkin-model in G of size at most $b + r \leq f(c)$. Thus we may assume

$$(10) \quad \mu(K) < c.$$

We have $\deg_K^*(v_C) \geq 1$ for every $C \in \mathcal{C}$. Let us say that a subgraph $C \in \mathcal{C}$ is *bad* if $\deg_K^*(v_C) = 1$, and *good* otherwise.

We color the vertices of each multipath $P \in \mathcal{P}$ as follows: a vertex $v \in V(P)$ is colored *black* if, in the graph G , all its neighbors outside P belong to bad subgraphs of \mathcal{C} ; the vertex v is colored *white* otherwise. (We remark that v could possibly have no neighbor outside P , in which case v is colored black by our definition.)

Claim 4. *If some multipath $P \in \mathcal{P}$ contains $(4c)^{4c}$ consecutive black vertices, then one can find a small c -pumpkin-model in G .*

Proof. Let Q be the subgraph of P induced by these $(4c)^{4c}$ black vertices. Let \mathcal{C}' be the subset of subgraphs $C \in \mathcal{C}$ such that v_C is adjacent to an *internal* vertex of Q in the graph J . Let $S := \{v_C : C \in \mathcal{C}'\}$. Since all vertices of Q are colored black, it follows that internal vertices of Q are only adjacent in J to vertices in $V(P) \cup S$, and that every subgraph $C \in \mathcal{C}'$ is bad.

Let H be the graph obtained from $J[V(P) \cup S]$ by contracting every edge of P not included in Q . Since every subgraph $C \in \mathcal{C}'$ is bad, it follows from (9) that, in H , every vertex in S has at least two neighbors in Q . Hence (H, Q) is a hedgehog of size $|Q| = (4c)^{4c}$.

The graph H is a minor of the subgraph G^* of G induced by

$$V(P) \cup \bigcup \{V(C) : C \in \mathcal{C}'\}.$$

Since vertices of P have degree at most k in G and $|C| \leq b$ for every $C \in \mathcal{C}'$, we have

$$(11) \quad |G^*| \leq r + r(k-1)b \leq f(c).$$

It follows that every c -pumpkin-model of H can be extended to a small c -pumpkin-model of G .

Applying Lemma 4.8 on (H, Q) , we obtain either a bad cutset X of (H, Q) or a c -pumpkin-model of H . We have seen that we are done in the latter case, so assume the former holds and let $\{u, v\} =: X$. Consider a connected component T of $H \setminus X$ that avoids both endpoints of Q and such that $|T| \geq 2$. Let Z be the subgraph of G induced by $(V(T) \cap V(Q)) \cup \bigcup \{V(C) : v_C \in V(T)\}$. It follows from the definition of H and our choice of T that Z is a connected component of $G \setminus X$ with $|Z| \geq 2$ and such that u and v are both adjacent to some vertex in Z . Thus (Z, u, v) is an outgrowth of G . This implies that $\Gamma := \Gamma(Z, u, v)$ has a c -pumpkin-model, because otherwise we could have

applied **Z2** on G with the outgrowth (Z, u, v) . Since Γ is a minor of G^* , by (11) this model can be extended to a small c -pumpkin-model of G . \square

By Claim 4, we may assume that, for every $P \in \mathcal{P}$, the number $w(P)$ of white vertices in P satisfies

$$(12) \quad w(P) \geq \frac{r}{(4c)^{4c}} = k.$$

Our aim now is to use (12) to define a minor of K with large minimum degree. First, for every good subgraph $C \in \mathcal{C}$, “assign” v_C to an arbitrary neighbor of v_C in K . Next, for every $w \in W$, contract all edges $v_C w$ of K into the vertex w for all vertices v_C assigned to w . Similarly, for every $P \in \mathcal{P}$, contract all edges $v_C v_P$ into the vertex v_P for all vertices v_C assigned to v_P . Finally, remove the vertex v_C for every bad subgraph $C \in \mathcal{C}$. The resulting graph is denoted L .

For every vertex of L there is a natural induced subgraph of G that corresponds to it, namely the subgraph defined by all the edges that were contracted into w . Let S_w and S_P be the (induced) subgraph of G that corresponds to the vertex $w \in W$ and v_P ($P \in \mathcal{P}$) of L , respectively. The subgraphs S_w ($w \in W$) and S_P ($P \in \mathcal{P}$) of G have diameter at most $2r$ and $3r$, respectively. Thus, by Lemma 2.1, a c -pumpkin-model of L of size q can be turned into one of G of size at most $3rc \cdot q$. Hence, in order to conclude the proof, it is enough to find a c -pumpkin-model in L of size at most $h(c) \log |L|$, since

$$h(c) \log |L| \leq \frac{f(c)}{3rc} \log |L| \leq \frac{f(c)}{3rc} \log n.$$

To do so, we will show that L has large minimum degree.

First consider a vertex $w \in W$. Let a be the number of edges incident with w in K such that the other endpoint is a vertex of the form v_C ($C \in \mathcal{C}$) that was assigned to w . By (9), w cannot be adjacent in K to a vertex v_C corresponding to a bad subgraph $C \in \mathcal{C}$. Thus, it follows from the definitions of good subgraphs and L that

$$\deg_L(w) \geq \frac{a}{\mu(K)} + (\deg_K(w) - a).$$

(The $\frac{a}{\mu(K)}$ term above comes from the fact that each vertex v_C that was assigned to w contributes at least one to the degree of w in L , while in K there were at most $\mu(K)$ edges between v_C and w .) Using (10) we obtain

$$(13) \quad \deg_L(w) > \frac{a}{c} + (\deg_K(w) - a) \geq \frac{\deg_K(w)}{c} \geq \frac{k}{c}.$$

Now consider a multipath $P \in \mathcal{P}$. Let a' be the number of edges incident with v_P in K such that the other endpoint is a vertex of the form v_C ($C \in \mathcal{C}$) that was assigned to v_P . Let b' be the number of edges incident with v_P in K that are not of the previous form and also not incident with a vertex v_C such that C is bad. By the definition of white vertices, we have $a' + b' \geq w(P)$ (recall that $w(P)$ is the number of white vertices in P). Using (12), it follows

$$(14) \quad \deg_L(v_P) \geq \frac{a'}{\mu(K)} + b' > \frac{a'}{c} + b' \geq \frac{w(P)}{c} \geq \frac{k}{c}.$$

It follows from (13) and (14) that L has minimum degree at least k/c . If $\mu(L) \geq c$, then L has a c -pumpkin-model of size two and we are trivially done, so let us assume $\mu(L) < c$. Then the underlying simple graph L' of L has minimum degree at least $k/c^2 \geq 2^{2\sqrt{c}+1}$. Using Corollary 4.10 on L' , we find a c -pumpkin-model in L of the desired size, that is, of size at most $h(c) \log |L|$.

Finally, we note that each step of the proof can easily be realized in polynomial time. Therefore, a small c -pumpkin-model of G can be found in polynomial time. \square

4.4. Algorithmic consequences. Lemma 4.11 can be used to obtain $\mathcal{O}(\log n)$ -approximation algorithms for both the MINIMUM c -PUMPKIN-COVERING and the MAXIMUM c -PUMPKIN-PACKING problems for every fixed $c \geq 1$, as we now show.

Algorithm 1 A $\mathcal{O}(\log n)$ -approximation algorithm.

INPUT: An arbitrary graph G

OUTPUT: A c -packing \mathcal{M} of G and a c -hitting set X of G s.t. $|X| \leq (f(c) \log |G|) \cdot |\mathcal{M}|$
 $\mathcal{M} \leftarrow \emptyset$; $X \leftarrow \emptyset$

If $|G| \leq 1$: Return \mathcal{M}, X /* G cannot have a c -pumpkin-minor */

Else:

If G is not c -reduced:

 Apply a reduction rule on G , giving a graph H

 Call the algorithm on H , giving a packing \mathcal{M}' and a c -hitting set X' of H

 Compute using Lemma 4.1(b) a c -packing \mathcal{M} of G with $|\mathcal{M}| = |\mathcal{M}'|$

 Compute using Lemma 4.1(a) a c -hitting set X of G with $|X| \leq |X'|$

 Return \mathcal{M}, X

Else:

 Compute using Lemma 4.11 a c -pumpkin-model $M = \{A, B\}$ of G with

$|A \cup B| \leq f(c) \log |G|$

$H \leftarrow G \setminus (A \cup B)$

 Call the algorithm on H , giving a packing \mathcal{M}' and a c -hitting set X' of H

$\mathcal{M} \leftarrow \mathcal{M}' \cup \{M\}$

$X \leftarrow X' \cup A \cup B$

 Return \mathcal{M}, X

Theorem 4.12. *Given an n -vertex graph G , a $\mathcal{O}(\log n)$ -approximation for both the MINIMUM c -PUMPKIN-COVERING and the MAXIMUM c -PUMPKIN-PACKING problems on G can be computed in polynomial time using Algorithm 1, for any fixed integer $c \geq 1$.*

Proof. Consider Algorithm 1, where $f(c)$ is the function in Lemma 4.11. We will show that this algorithm provides a $\mathcal{O}(\log n)$ -approximation for the two problems under consideration.

It should be clear that the collection \mathcal{M} returned by Algorithm 1 is a c -packing of G , and similarly that the set X is a c -hitting set of G . Thus it is enough to show that they satisfy $|X| \leq (f(c) \log |G|) \cdot |\mathcal{M}|$ as claimed in the description of the algorithm. Indeed, since $|\mathcal{M}| \leq \nu_c(G) \leq \tau_c(G) \leq |X|$ and $f(c)$ is a constant depending only on c , this implies that the approximation factor of Algorithm 1 is $\mathcal{O}(\log n)$ for both the MINIMUM c -PUMPKIN-COVERING and the MAXIMUM c -PUMPKIN-PACKING problems.

We prove the inequality $|X| \leq (f(c) \log |G|) \cdot |\mathcal{M}|$ by induction on $|G|$. The inequality is obviously true in the base case, namely when $|G| \leq 1$, so let us assume $|G| > 1$.

If G is not c -reduced, then by induction the packing \mathcal{M}' and the c -hitting set X' of H considered by the algorithm satisfy $|X'| \leq (f(c) \log |H|) \cdot |\mathcal{M}'|$, and we obtain

$$|X| \leq |X'| \leq (f(c) \log |H|) \cdot |\mathcal{M}'| = (f(c) \log |H|) \cdot |\mathcal{M}| \leq (f(c) \log |G|) \cdot |\mathcal{M}|$$

as desired.

If G is c -reduced, then by induction the packing \mathcal{M}' and the c -hitting set X' of H satisfy $|X'| \leq (f(c) \log |H|) \cdot |\mathcal{M}'|$, and we have

$$\begin{aligned} |X| &= |X'| + |A \cup B| \\ &\leq (f(c) \log |H|) \cdot |\mathcal{M}'| + f(c) \log |G| \\ &\leq (f(c) \log |G|) \cdot (|\mathcal{M}'| + 1) \\ &= (f(c) \log |G|) \cdot |\mathcal{M}|. \end{aligned}$$

Thus $|X| \leq (f(c) \log |G|) \cdot |\mathcal{M}|$ holds in all cases.

Finally, we observe that there are at most n recursive calls during the whole execution of the algorithm, which implies that its running time is polynomial in n . \square

We would like to conclude by observing that Lemma 4.11 also implies the existence of an algorithm for p - c -COVER with running time $2^{\mathcal{O}(k \log \log k)} \cdot n^{\mathcal{O}(1)}$. While it is faster than the $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ algorithm that follows directly from the kernel of size $\mathcal{O}(k^2 \log^{3/2} k)$ given by [18], it is of course not as good as the one presented in Section 3. Nevertheless, we find it interesting that a running time of $2^{\mathcal{O}(k \log \log k)} \cdot n^{\mathcal{O}(1)}$ can already be achieved without using techniques such as iterative compression.

Theorem 4.13. *There exists an algorithm to solve the p - c -COVER problem in time $2^{\mathcal{O}(k \log \log k)} \cdot n^{\mathcal{O}(1)}$.*

Proof. We only provide a rough sketch of the proof; the interested reader should have no problem filling in the missing details.

First we compute the mentioned kernel K of size $\mathcal{O}(k^2 \log^{3/2} k)$. Then we run the following recursive algorithm on K :

- Apply iteratively reduction rules **Z1** and **Z2** on K until K is c -reduced.
- Compute using Lemma 4.11 a c -pumpkin-model M of K of size $\mathcal{O}(\log(k^2 \log^{3/2} k)) = \mathcal{O}(\log k)$.
- Branch on every vertex of M and repeat this procedure recursively.

Note that, in the last step of the algorithm, every solution must contain at least one vertex of M . Hence this yields a parameterized algorithm with running time $\mathcal{O}(\log^k k) \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log \log k)} \cdot n^{\mathcal{O}(1)}$. \square

5. CONCLUDING REMARKS

On the one hand, we provided an FPT algorithm running in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ deciding, for any fixed $c \geq 1$, whether all c -pumpkin-models of a given graph can be covered by at most k vertices. In our algorithms we used protrusions, but it may be possible to avoid it by further exploiting the structure of the graphs during the iterative compression routine (for example, a graph excluding the 3-pumpkin is a forest of cacti). We did not focus on optimizing the constants involved in our algorithms; it may be worth doing it, as well as enumerating all solutions, in the same spirit of [20] for FEEDBACK VERTEX SET.

It is natural to ask whether there exist faster algorithms for sparse graphs. Also, it would be interesting to have lower bounds for the running time of parameterized algorithms for this problem, in the spirit of those recently provided in [24]. One could as well consider other containment relations, like topological minor, induced minor, or contraction minor.

A more difficult problem seems to find single-exponential algorithms for the problem of deleting at most k vertices from a given graph so that the resulting graph has tree-width bounded by some constant. One could also consider the parameterized version of packing disjoint c -pumpkin-models, as it has been done for $c = 2$ in [5].

On the other hand, we provided a $\mathcal{O}(\log n)$ -approximation for the problems of packing the maximum number of vertex-disjoint c -pumpkin-models, and covering all c -pumpkin-models with the smallest number of vertices. It may be possible that the covering version admits a constant-factor approximation; so far, such an algorithm is only known for $c \leq 3$.

As mentioned in the introduction, for the packing version there is a lower bound of $\Omega(\log^{1/2-\varepsilon} n)$ on the approximation ratio (under reasonable complexity-theoretic assumptions). In fact, this lower bound applies to both the vertex-disjoint packing and the edge-disjoint packing [19]. For $c = 2$, the problem of packing a maximum number of edge-disjoint cycles admits a $\mathcal{O}(\sqrt{\log n})$ -approximation, whereas up to date $\mathcal{O}(\log n)$ is the best approximation ratio known for vertex-disjoint cycles [22]. Therefore, one might expect to get better approximation algorithms for packing edge-disjoint c -pumpkin models.

Finally, a class of graphs \mathcal{H} has the *Erdős-Pósa property* if there exists a function f such that, for every integer k and every graph G , either G contains k vertex-disjoint subgraphs each isomorphic to a graph in \mathcal{H} , or there is a set $S \subseteq V(G)$ of at most $f(k)$ vertices such that $G \setminus S$ has no subgraph in \mathcal{H} . Given a connected graph H , let $\mathcal{M}(H)$ be the class of graphs that can be contracted to H . Robertson and Seymour [27] proved that $\mathcal{M}(H)$ satisfies the Erdős-Pósa property if and only if H is planar. Therefore, for every $c \geq 1$, the class of graphs that can be contracted to the c -pumpkin satisfies the Erdős-Pósa property. But the best known function f is super-exponential (see [12]), so it would be interesting to find a better function for this case. The only known lower bound on f is $\Omega(k \log k)$ when $c \geq 2$, which follows from the $\Omega(k \log k)$ lower bound given by Erdős and Pósa in their seminal paper [14] for $c = 2$.

REFERENCES

- [1] V. Bafna, P. Berman, , and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.
- [2] R. Balasubramanian, M. Fellows, and V. Raman. An improved fixed parameter algorithm for Vertex Cover. *Information Processing Letters*, 65:163–168, 1998.
- [3] A. Becker and D. Geiger. Optimization of pearl’s method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83:167–188, 1996.
- [4] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) kernelization. In *Proc. of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 629–638, 2009.
- [5] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. In *Proc. of the 17th Annual European Symposium on Algorithms (ESA)*, volume 5757 of *LNCS*, pages 635–646, 2009.
- [6] H. L. Bodlaender, J. van Leeuwen, R. B. Tan, and D. M. Thilikos. On interval routing schemes and treewidth. *Information and Computation*, 139(1):92–109, 1997.
- [7] Y. Cao, J. Chen, and Y. L. 0002. On feedback vertex set new measure and new structures. In *Proc. of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 6139 of *LNCS*, pages 93–104, 2010.
- [8] J. Chen, B. Chor, M. Fellows, X. Huang, D. W. Juedes, I. A. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216–231, 2005.
- [9] J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.
- [10] J. Chen, Y. Liu, S. Lu, B. O’Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. In *Proc. of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 177–186, 2008.
- [11] F. K. H. A. Dehne, M. R. Fellows, M. A. Langston, F. A. Rosamond, and K. Stevens. An $\mathcal{O}(2^{O(k)} n^3)$ FPT Algorithm for the Undirected Feedback Vertex Set Problem. *Theory of Computing Systems*, 41(3):479–492, 2007.
- [12] R. Diestel. *Graph Theory*, volume 173. Springer-Verlag, 2005.
- [13] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer-Verlag, 1999.

- [14] P. Erdős and L. Pósa. On independent circuits contained in a graph. *Canadian Journal of Mathematics*, 17:347–352, 1965.
- [15] S. Fiorini, G. Joret, and U. Pietropaoli. Hitting Diamonds and Growing Cacti. In *Proc. of the 14th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 6080 of *LNCS*, pages 191–204, 2010.
- [16] S. Fiorini, G. Joret, D. O. Theis, and D. R. Wood. Small Minors in Dense Graphs. Manuscript, available at <http://arxiv.org/abs/1005.0895>, 2010.
- [17] F. V. Fomin, S. Gaspers, D. Kratsch, M. Liedloff, and S. Saurabh. Iterative compression and exact algorithms. *Theoretical Computer Science*, 411(7-9):1045–1053, 2010.
- [18] F. V. Fomin, D. Lokshtanov, N. Misra, G. Philip, and S. Saurabh. Hitting forbidden minors: Approximation and Kernelization. In *Proc. of the 28th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *LIPICs*, pages 189–200, 2011.
- [19] Z. Friggstad and M. Salavatipour. Approximability of packing disjoint cycles. *Algorithmica*, 60:395–400, 2011.
- [20] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396, 2006.
- [21] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
- [22] M. Krivelevich, Z. Nutov, M. R. Salavatipour, J. Yuster, and R. Yuster. Approximation algorithms and hardness results for cycle packing problems. *ACM Transactions on Algorithms*, 3(4), 2007.
- [23] M. Lampis. Algorithmic meta-theorems for restrictions of treewidth. In *Proc. of the 18th Annual European Symposium on Algorithms (ESA)*, volume 6346 of *LNCS*, pages 549–560, 2010.
- [24] D. Lokshtanov, D. Marx, and S. Saurabh. Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal. In *Proc. of the 22nd annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 777–789, 2011.
- [25] B. A. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.
- [26] N. Robertson and P. Seymour. Graph Minors. XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.
- [27] N. Robertson and P. D. Seymour. Graph Minors. V. Excluding a Planar Graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986.

DÉPARTEMENT D'INFORMATIQUE
 UNIVERSITÉ LIBRE DE BRUXELLES
 BRUSSELS, BELGIUM
E-mail address: `gjoret@ulb.ac.be`

ALGCo PROJECT-TEAM
 CNRS, LIRMM
 MONTPELLIER, FRANCE
E-mail address: `{paul,sau,thomasse}@lirmm.fr`

THE INSTITUTE OF MATHEMATICAL SCIENCES
 CHENNAI, INDIA
E-mail address: `saket@imsc.res.in`

